IMPLEMENTASI METODE *EXTREME LEARNING MACHINE* (ELM) UNTUK MEMPREDIKSIKAN PENJUALAN ROTI (STUDI KASUS : HARUM *BAKERY*)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Luqman Hakim Harum
NIM: 135150207111104

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI METODE *EXTREME LEARNING MACHINE* (ELM) UNTUK MEMPREDIKSIKAN PENJUALAN ROTI (STUDI KASUS : HARUM *BAKERY*)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Luqman Hakim Harum
NIM: 135150207111104

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Nurul Hidayat, S.Pd., M.Sc	Ratih Kartika Dewi, S.T., M.Kom
NIP: 196804302002121001	NIK: 2015038905202001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518200312001

i

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 14 Mei 2018

Luqman Hakim Harum
NIM: 135150207111104

ii

KATA PENGANTAR

Segala puji syukur atas kehadirat Allah SWT yang elalu penulis ucapkan, yang telah memberikan rahmat dan hidayahNya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Metode *Extreme Learning Machine*(ELM) untuk Memprediksi Penjualan Roti (Studi Kasus : Harum *Bakery*)".

Penulis juga ingin menyampaikan banyak terimakasih kepada pihak-pihak yang telah membantu dan mensupport penelitu dalam penyelesaian skripsi ini yang diantaranya adalah:

1. Nurul Hidayat, S.Pd., M,SC selaku dosen pembimbing 1 yang telah meluangkan cukup waktu, memberi masukan positif dan selalu mendukung pengerjaan skripsi ini.
2. Ratih Kartika Dewi, S.T., M.Kom selaku dosen pembimbing 2 yang telah banyak memberikan masukan positif, meluangkan waktu, dan mendukung pengerjaan skripsi demi terselesaikannya skripsi ini.
3. Seluruh dosen FILKOM Universitas Brawijaya atas ketersediannya sebagai pemberi ilmu selama masa perkuliahan.
4. Ayah, ibu, serta saudara dirumah yang selalu mendoakan dan memberi dukungan baik finansial maupun moral selama ini.
5. Trika Reyza Palevi, Rudi Rido Romansyah, Ayustina Gusti, Rimba Kurniawan, Group Battalion Edo Sempol, Group Konco Dulur dan seluruh mahasiswa FILKOM 2013 yang telah memberikan semangat serta dukungan.
6. Seluruh pihak yang membantu terselesaikannya skripsi ini yang tidak bisa penulis sebutkan satu persatu.

Pada penulisan skripsi ini penulis sadar masih banyak kekurangan dan jauh dari kata sempurna sehingga penulis sangat bertrimakasih jika ada pihak yang memberi masukan untuk lebih menyempurnakan penelitian ini. Selain itu penulis juga ingin mengucapkan terimakasih bila selanjutnya ada penelitian yang mengembangkan lebih jauh dari penelitian ini.

Malang, 14 Mei 2018

Penulis

Hakim.harum@gmail.com

ABSTRAK

Harum *Bakery* merupakan sebuah toko roti yang terletak di daerah Kabupaten Malang. Jumlah penjualan roti pada perusahaan ini tidak menentu setiap harinya. Hal ini membuat perusahaan kesulitan untuk memprediksi penjualan roti perharinya. Agar tidak mengalami kerugian, perusahaan membutuhkan sistem untuk mempermudah prediksi penjualan. Dengan prediksi penjualan roti tersebut, harapannya perusahaan bisa menekan kerugian yang mungkin terjadi dan mengoptimalkan keuntungan. Penelitian ini menerapkan metode *Extreme Learning Machine* (ELM) yang merupakan metode dari Jaringan Syaraf Tiruan (JST) untuk memprediksi penjualan roti pada Harum *Bakery*. Proses dari prediksi menggunakan metode ELM ini adalah dimulai dari normalisasi data, kemudian proses *training*, proses *testing*, mencari perhitungan dari nilai *error* menggunakan metode *Mean Square Error* (MSE) untuk mencari nilai *error* terkecil dengan beberapa pengujian dan kemudian melakukan denormalisasi data. Metode ELM merupakan metode *feedforward* dengan *single hidden layer* yang kemudian disebut dengan *Single Hidden Layer Feedforward Neural Network* (SLFNs). Tujuan utama dari dibuatnya metode ini adalah ditujukan untuk memperbaiki kelemahan-kelemahan dari jaringan syaraf tiruan *feedforward* lainnya terutama dalam *learning speed*. Berdasarkan beberapa pengujian yang telah dilakukan, maka penelitian menghasilkan tingkat *error* terkecil sebesar 0,01616 untuk roti tawar dengan menggunakan 7 *neuron* pada *hidden layer,* 4 fitur, dan data penjualan selama 5 bulan, nilai MSE terbaik sebesar 0,02839 untuk roti manis dengan menggunakan 2 *neuron* pada *hidden layer, 5 fitur*, dan 4 bulan penjualan roti, dan 0,00812 untuk roti cake dengan menggunakan 7 *neuron* pada *hidden layer,* 4 fitur, dan 3 bulan penjualan roti.

Kata kunci: *prediksi penjualan, extreme learning machine, jaringan syaraf tiruan, mean square error, single hidden layer feedforward neural network.*

ABSTRACT

Harum Bakery is a bread store located in Malang Regency area. The number of bread sales in this company is uncertain everyday. It makes the company difficult to predict the sale of breads per day. To avoid loss, this company need a system to predict sales prediction easily. With the prediction of the sale, the writer hope that the company can suppress the losses that may occur and optimizing company's profit. This research use Extreme Learning Machine (ELM) method which is method of Artificial Neural Network(ANN) to predict bread sales at Harum Bakery. The Process of prediction using ELM method is started from data normalization, then training process, testing process, find the error value using Mean Square Error (MSE) method to find the smallest error value with some testing, and data denormalization the ELM method is feedforward method with a single hidden layer which is called Single Hidden Layer Feedforward Neural Network (SLFNs). The main purpose of this method is to improve the weakness of other feedforward artificial neural networks, especially in the learning speed. Based on some tests that have been done, the smallest error rate is 0,01616 for white bread using 7 neurons, 4 features, and 5 months of sales data, the best MSE is 0,02839 for sweet bread using 2 neurons, 5 features, and 4 months of sales data, and 0,00812 for cake bread using 7 neurons, 4 features, and 3 months of sales data.

Keywords: *sales prediction, extreme learning machine, artificial neural network, mean square error, single hidden layer feedforward neural network.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I PENDAHULUAN

1.1 Latar Belakang

Di era modern ini, perkembangan ilmu teknologi dan ilmu komputer sangatlah diperlukan baik itu dalam hal komputasi, prediksi, klasisfikasi, dan masih banyak hal lagi yang dapat membantu manusia baik itu dalam bidang pendidikan, perkonomian, industri, bisnis, pariwisata, militer, dan dalam bidang bidang yang lain tentunya.

Roti sekarang sudah menjadi makanan yang sudah sangat digemari oleh seluruh kalangan di Indonesia baik itu roti manis, roti tawar, maupun roti cake. Tak heran lagi kalau roti sekarang sudah menjadi bagian dari kehidupan masyarakat sehari hari, dikarenakan harga yang murah dan mudah untuk dijangkau. Tidak jarang orang mengubah pola makannya dengan mengkonsumsi roti sebagai pengganti makanan sehari hari, mulai dari sarapan, makan siang, hingga makan malam.

Harum *Bakery* merupakan sebuah toko roti yang terletak di daerah Kabupaten Malang, tepatnya di Jalan Raya Sengkaling No 217. Harum *Bakery* memproduksi roti tidak hanya untuk dijual, namun juga disumbangkan setiap minggunya ke panti asuhan dan masjid-masjid sekitar toko Harum *Bakery*. Perusahaan ini menjual roti dengan jumlah penjualan yang variatif setiap harinya. Sekitar 5 tahun yang lalu, perusahaan memproduksi roti untuk penjualan di toko dengan jumlah yang besar yaitu memproduksi sebanyak 5kg masing-masing jenis roti untuk pertama kali membuat dalam hari itu. Untuk 5kg bahan roti masing-masing menghasilkan sekitar 240 roti manis, 20 roti tawar, dan 35 roti cake. Ketika jumlah permintaan lebih sedikit dari produksi dan setiap hari perusahaan memproduksi sebanyak 5kg untuk pertama kali membuat dalam hari itu, maka terjadi penumpukan roti. Hal ini nantinya berujung pada roti diturunkan dari toko karena kondisi roti sudah tidak layak untuk di jual. Kemudian kini perusahaan telah mengubah jumlah roti yang di produksi dengan jumlah yang lebih kecil, yaitu 3kg bahan roti yang menghasilkan sekitar 135 roti manis, 12 roti tawar, dan 21 roti cake. Demikian juga, ketika jumlah permintaan lebih besar dari produksi, toko kehabisan roti untuk dijual, maka perusahaan harus membuat roti lagi. Hal ini merugikan perusahaan, karena untuk membuat roti lagi, perusahaan harus rela membuang tenaga, listrik, gas, waktu, dan lain lain. Dengan naik turunnya penjualan roti setiap harinya, hal ini membuat perusahaan kesulitan untuk memprediksi penjualan roti perharinya. Karena jika perusahaan bisa memprediksi penjualan roti, perusahaan dapat mengoptimalkan produksi roti untuk penjualan di toko.

Penelitian sebelumnya yang memiliki topik yang hampr sama yang diteliti oleh Sukmarani (2016), yaitu mengenai *forecasting* berjudul "Penerapan Metode *Exponential Smoothing* pada Peramalan Penjualan Dalam Penentuan kuantitas Produksi roti" berdasarkan data dan hasil perhitungan dari *exponential*

smoothing tersebut, didapatkanlah nilai MSE terkecil alpha 0,2 sebesar 586,74 dan memiliki tingkat keakuratan sebesar 68,852% dengan data latih yang digunakan adalah selama 2 minggu penjualan roti (Sukmarini, Statiswaty, & Ramadhan, 2016).

Pada penelitian sebelumnya yang juga hampir mirip yang ditulis oleh Khairia Istiqara yaitu mengenai prediksi suatu kebutuhan air yang berjudul "Prediksi Kebutuha Air PDAM Kota Malang Menggunakan Metode *Fuzzy Time Series* dengan Algoritma Genetika". Berdasarkan data dan hasil perhitungan dari metode *Fuzzy Time Series* dan algoritma genetika manghasilkan nilai eror(MAPE) sebesar 2.266776% yang dihasilkan dari proses algoritma genetika. Dengan hasil tersebut, maka optimasi menggunaka Fuzzy Time Series dan Algoritma Genetika ini menghasilkan prediksi yang cukup akurat dengan nilai MAPE sebesar 2.266776% (Istiqara, 2017).

Dalam ilmu informatika, banyak metode yang dapat kita gunakan untuk melakukan perediksi terhadap sesuatu. Pada penelitian kali ini, peneliti menggunakan salahsatu metode yang merupakan metode pembelajaran baru dari ilmu Jaringan Syaraf Tiruan (JST) yaitu metode *Extreme Learning Machine* atau yang biasa disebut dengan ELM. Metode ini pertama kali ditemukan oleh Huang pada tahun 2004. Metode *Extreme Learning Machine* merupakan jaringan syaraf tiruan *feedforward* dengan *single hidden layer* yang kemudian disebut dengan *Single Hidden Layer Feedforward neural Network* (SLFNs). Tujuan utama dari dibuatnya metode ini adalah ditujukan untuk memperbaiki kelemahan kelemahan dari jaringan syaraf tiruan feedforward lainnya terutama dalam *Learning* Speednya (Huang, Zhu , & Siew, Extreme Learning Machine : a New Learning Scheme of Feedforward Neuralnetworks, 2004).

Pada penelitian yang memiliki metode yang sama adalah penelitian dengan judul "Klasifikasi Penyimpangan Tumbuh Kembang Anak Menggunakan Metode *Extreme Learning Machine* (ELM)" yang diteliti oleh Makrina Christy Ariestyani, menghasilkan nilai akurasi 76,67% dengan menggunakan fungsi sigmoid serta menggunakan 10 buah *hidden neuron*. Dengan menggunakan metode *Extreme Learning Machine* atau ELM ini, hasil memaparkan bahwa kelas normal dan ADHD memiliki akurasi data yang baik, sedangkan kelas DS dan autis menghasilkan hasil yang tidak cukup baik (Arestyani, 2017).

Berdasarkan latar belakang yang telah penulis tuliskan, penulis berkeinginan untuk menulis sebuah penelitian dengan judul "Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*)" yang bertujuan untuk mengaplikasikan metode ELM ini untuk memprediksikan penjualan roti di Harum *Bakery*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah penulis paparkan, maka penulis merumuskan beberapa rumusan masalah yang akan diselesaikan dalam penelitian ini yaitu :

1. Bagaimana mengimplementasikan metode ELM untuk prediksi penjualan roti pada Harum *Bakery*.
2. Berapa hasil tingkat error atau kesalahan dari prediksi penjualan roti menggunakan metode ELM dan melalui perhitungan *Mean Square Error* (MSE).

1.3 Tujuan Penelitian

Tujuan penelitian dari penelitian ini adalah :

1. mengimplementasikan metode *Extreme Learning Machine* (MSE) untuk melakukan prediksi pada penjualan roti di Harum bakery.
2. Menghitung nilai tingkat kesalahan prediksi penjualan roti di Harum Bakery menggunakan Mean Square *Error* (MSE).

1.4 Batasan Masalah

Untuk menghindari pembahasan yang terlalu luas, maka dari rumusan masalah penulis membatasi penelitian ini dengan ketentuan sebagai berikut :

1. Ada tiga jenis roti yang peneliti gunakan, yaitu roti manis, roti tawar, dan roti cake.
2. Data yang digunakan dalam penelitian ini adalah data penjualan roti per hari pada perusahaan Harum *Bakery* selama 5 bulan, periode September 2017-Januari 2017, yang didapatkan dengan meminta data penjualan secara langsung kepada *owner* dari Harum *Bakery* dengan jumlah total 153 data penjualan roti manis, 153 data penjualan roti tawar, dan 153 data penjualan roti cake.
3. Fungsi aktivasi yang digunakan oleh peneliti adalah fungsi aktivasi sigmoid.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk membantu memecahkan permasalaahan yang dialami oleh Harum *Bakery* pada latar belakang, yaitu dengan mempermudah perusahaan dalam melakukan prediksi penjualan roti pada Harum *Bakery*. Sehingga harapannya, perusahaan dapat memprediksi penjualan roti dengan lebih mudah dan lebih tepat. Dengan prediksi yang lebih tepat tersebut, harapannya perusahaan kemudian dapat menentukan jumlah produksi roti perhari dengan tepat, kemudian peneliti berharap perusahaan dapat mengurangi kerugian dan mengoptimalkan keuntungan dalam penjualan roti di Harum *Bakery*.

1.6 Metodologi Penulisan

Sistematika penulisan yang diterapkan pada penelitian ini adalah sebagai berikut:

BAB I Pendahuluan

Berisi tentang uraian umum yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat bagi peneliti dan perusahaan, dan metodologi penulisan laporan penelitian ini.

BAB II Tinjauan Pustaka

Membahas tentang teori-teori yang dugunakan dalam peneltian serta kajian pustaka yang mendukung metode yang kami gunakan demi terwujidnya Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti dengan studi kasus Harum *Bakery*.

BAB III Metodologi

Membahas tentang metode-metode yang digunakan dalam penelitian Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*).

BAB IV Perancangan

Membahas tentang analisa, arsitektur sistem, perhitungan manualisasi, perancangan antarmuka, dan pengujiaan algoritma dari Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*).

BAB V Implementasi

Membahas tentang implementasi sistem dan implementasi antarmuka pada Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*).

BAB VI Pengujian

Memaparkan tentang pengujian dari Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*) yang kemudian menggunakan *Mean Square Error* (MSE) sebagai tolak ukurnya.

BAB VII Penutup

Memaparkan kesimpulan dari penelitian Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*). yang telah diselesaikan, serta saran dari penulis untuk pengembang yang ingin mengembangkan penelitian ini lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tentang kajian pustaka berupa referensi-referensi penelitian sebelumnya yang mendukung penelitian ini serta berisi tentang penjelasan konsep dasar teori yang penulis gunakan pada penelitian ini. Konsep dasar teori yang penulis pakai adalah *Extreme Learning Machinel* (ELM).

2.1 Kajian Pustaka

Kajian pustaka ini membahas tentang sumber sumber referensi yang relevan yaitu mengenai prediksi dengan objek yang berbeda dan dengan metode yang sama. Tabel 2.1 merupakan tabel penelitian sebelumnya yang dibuat sebagai referensi atau rujukan dalam penunjang penulisan penelitian ini.

Tabel 2.1 Tabel Penelitian Sebelumnya

No.	Judul Penelitian	Nama Penulis	Input	Metode	Hasil
1	Klasifikasi Penyimpangan Tumbuh Kembang Anak Menggunakan Metode *Extreme Learning Machine* (ELM)	(Arestyani, 2017)	- data gejala gejala penyimpangan tumbuh kembang anak. - Kuisioner user.	*Extreme Learning Machine (ELM)*	Dengan metode ELM didapatkan hasil akurasi tertinggi yaitu 76,67% menggunakan fungsi aktivasi sigmoid dan jumlah *hidden neuron 10 buah.* dengan menggunakan pengujian *confusion matrix* dengan perhitungan f-measure didapatkan akurasi tinggi data yang baik untuk normal dan ADHD, dan *down syndrome* dan autis memiliki nilai akurasi yang tidak cukup baik.
2	Sistem	(Fardani,	- Data	*Extreme*	Dengam metode

	Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode : *Extreme Learning Machine* (Studi Kasus : Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto)	Wuryanto, & Werdiningsih , 2015)	Jumlah Kunjungan Pasien. - Input parameter.	*Learning Machine (ELM)*	ELM didapatkan nilai MSE yang optimal yaitu sebesar 0.027 pada fungsi aktivasi *sigmoid biner* dengan jumlah *hidden layer* sejumlah 7 buah dan *Epoch* 500.
3	Prediksi Jumlah Permintaan Koran Menggunakan Metode *Extreme Learning Machine*	(Ramadhanty, 2016)	-data penjualan bulanan koran. -jumlah *hidden neuron.* -fungsi aktivasi. -perbandingan data latih dan uji.	*Extreme Learning Machine (ELM)*	Dengan metode ELM dan beberpa pengujian didapatkan tingkat MSE yang rendah sebesar 0.009311 yang mana pada penelitian ini menggunakan fungsi aktivasi sigmoid dengan 7 fitur, jumlah *neuron* pada *hidden layer* sebanyak 7, dan menggunakan perbandingan 80% data latih dan 20% data uji.
4	Jaringan Syaraf Tiruan *Extreme Learning Machine* (ELM) Untuk Memprediksi Kondisi Cuaca di Wilayah	(Humaini, 2015)	- data cuaca wilayah Malang selama 2014-2015. - Kecepatan angin. - Suhu	*Extreme Learning Machine (ELM)*	Dengan menggunakan metode ELM maka didapatkannya nilai MSE sebesar 0.064560051 dengan 4 *hidden*

	Malang		udara. - Kelembaban udara. - Tekanan udara.		*node* dan menggunakan bobot dan bias optimal.
5	Prediksi Penjualan Mi Menggunakan Metode *Extreme Learning Machine (ELM)* Di Kober Mie Setan Cabang Soekarno Hatta	(Gusti, 2017)	- Data historis penjualan kober mi setan. - Data historis stock kober mi setan.	*Extreme Learning Machine (ELM)*	Dengan menggunakan metode ELM menghasilkan nilai eror MSE terbaiknya adalah 0.0171

2.2 Dasar Teori

Dasar teori yang digunakan untuk menunjang penelitian skripsi ini adalah mengenai konsep dasar *Bakery,* konsep dasar prediksi, konsep dasar Jaringan Syaraf Tiruan (JST), konsep dasar metode *Extreme Learning Machine* (ELM), konsep dasar *min-max Normalization,* dan konsep dasar *Mean Square Error (*MSE*).*

2.2.1 Toko Roti (*Bakery*)

Kata Toko Roti dalam bahasa Inggris dikenal dengan sebutan *Bakery*. Ada beberapa jenis roti yang biasa dijual pada sebuah perusahaan *Bakey* yaitu roti manis, roti tawar, dan roti cake. *Bakery* dapat dibagi menjadi beberapa produk makanan yang akan dijual, yaitu :

- *Bakery* Tradisional, yaitu salah satu *bakery* yang memproduksi atau menjual roti-roti maupun kue kue tradisional yang sudah dikenal sebagai roti atau kue khas Indonesia.
- *Bakery Pastry*, *Bakery* model ini biasanya terdapat di tempat tempat mewah seperti hotel karena *Bakery* ini menjual atau memproduksi roti internasional, seperti *French Bread, Dinner atau Breakfast Roll, Ciabatta,* dan masih banyak lagi lainnya.
- *Bakery* Individual, adalah sebuah *Bakery* yang pemodalan awal berasal dari suatu indvidu bukan hasul dari kerjasama dari orang lain. biasanya *Bakery* individu ini hanya mengambil modal dari individu itu sendiria atau keluarga. Dan tidak mengikuti standar tertentu. Contoh *Bakery* yang termasuk dalam *Bakery* individual adalah Citra *Bakery,* Harum *Bakery,* Roti Boy dan masih banyak lagi.

- *Franchise Bakery, Bakery* ini memiliki aturan atau standar yang telah ditentukan oleh penanam modal, pada *Bakery* ini, semua hal yang ada didalamnya harus detail sesuai standar yang ada. Biasanya *Bakery* ini berasal dari luar negri seperti Bread Talk (Taiwan) dan lain sebagainya.

2.2.2 Prediksi

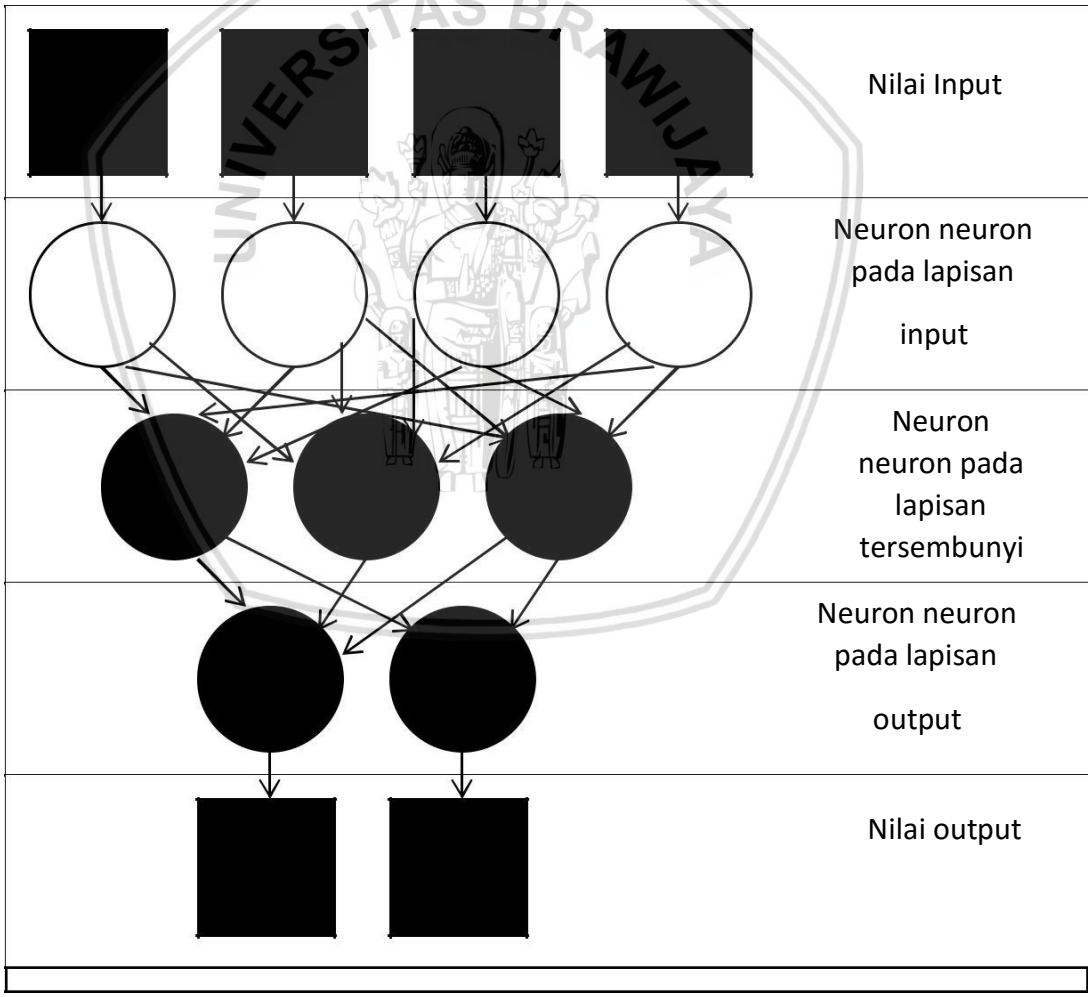
Prediksi merupakan suatu kegiatan yang bertujuan untuk memperkirakan suatu kejadian dimasa yang akan datang dengan menggunakan perhitungan-perhitungan yang berdasarkan data dan analisa baik itu ilmiah maupun non-ilmiah. Ada 3 jenis prediksi, yang pertama adalah prediksi jangka pendek, jangka panjang, dan jangka menengah. Prediksi jangka pendek adalah suatu prediksi yang dilakukan dengan menggunakan pola-pola yang ada pada data, dan membutuhkan waktu yang singkat terhadap perubahan berdasarkan faktor yang membentuk pola data. Sedangkan prediksi jangka panjang dan menengah adalah suatu prediksi yang digunakan untuk perencanaan strategis. Prediksi jangka panjang merupakan prediksi data *time-series* yaitu meramalkan suatu niai variable berdasarkan waktu tanpa mempedulikan faktor-faktor yang lain (Mate, Ferrandez, & Pereal, 2016).

2.2.3 Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan pertamakali didesign pada tahun 1943 oleh ilmuan yang bernama Warren McCulloch dan Walter Pitts. Konsep dasar dari pembuatan jaringan syaraf tiruan ini adalah menerapkan model-model matematik yang diterapkan seperti pada fungsi kerja pada otak/*neuron*. Metode yang dikembangkan seperti syaraf biologi ini adalah langkah maju di dunia Ilmu komputer (Kusumadewi, 2003).

Jaringan syaraf tiruan atau yang biasa disebut JST ini merupakan representasi dari kinerja otak manusia, seperti yang kita ketahui, otak manusia selalu mengalami masa pembelajaran. Dalam jaringan syaraf tiruan ini sama seperti otak manusia yang didalamnya terdapat neuron-neuron yang saling memiliki keterkaitan atau berhubungan satu dengan yang lain. informasi-informasi yang diterima oleh neuron akan dikirimkan dari satu neuron ke neuron yang lain. Pada jaringan syaraf, terdapat lapisan-lapisan *neuron* (*neuron layers*) yang lapisan tersebut memiliki koneksi dengan lapisan neuron sebelum atau sesudah lapisan itu sendiri. Tapi hal ini tidak berlaku terhadap lapisan input dan output. (Sinuhaji, 2009).

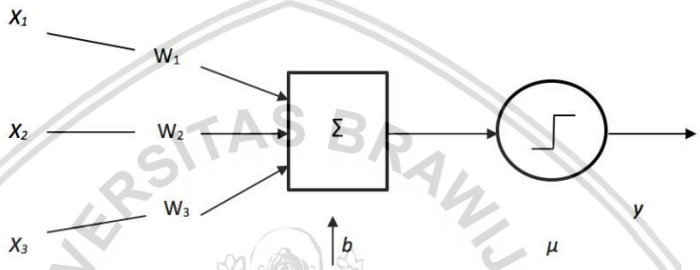
Ada beberapa jenis jaringan, yang pertama adalah jaringan dengan lapisan tunggal. Jaringan dengan lapisan tunggal ini hanya memiliki satu lapisan dengan bobot-bobot terhubung. Pada jaringan ini tidak terdapat lapisan *hidden layer*/lapisan tersembunyi. Dengan demikian, jaringan ini hanya menerima input yang kemudian diolah langsung menjadi output tanpa melewati *hidden layer*/lapisan tersembunyi. Yang kedua adalah jaringan dengan banyak lapisan. Pada jaringan ini, terdapat satu atau lebih lapisan *neuron*/*hidden neuron* antara input dan output. Umumnya lapisan-lapisan tersebut memiliki bobot yang terletak diantara 2 lapisan yang bersebelahan. Jaringan syaraf tiruan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit atau kompleks dibandingkan jaringan syaraf tiruan dengan lapisan tunggal. Gambar 2.1 menunjukkan JST dengan banyak lapisan dengan 3 *layer*/lapisan. (Sinuhaji,2009) Yang ketiga adalah jaringan dengan lapisan kompetitif. Pada jaringan dengan lapisan kompetitif ini, hubungan antar neuronnya biasanya tidak ditampilkan dalam arsitektur. Gambar 2.1 merupakan JST dengan 3 *layer*/lapisan (Kusumadewi, 2003).



Gambar 2.1 JST dengan 3 *layer*/lapisan. Sumber : (Sinuhaji, 2009)

Struktur dari Jaringan syaraf tiruan ini memiliki beberapa nilai input sebanyak n. Seperti pada Gambar 2.2 input tersebut adalah , , dan . Dari input tersebut maka input tersebut akan diproses oleh bobot , , dan . Semua

input tersebut kemudian akan diproses dengan suatu fungsi yang akan menjumlahkan semua nilai bobot yang datang. setelah itu hasil dari penjumlahan nilai-nilai bobot yang datang akan dibandingkan dengan suatu nilai *threshold* tertentu melalui fungsi aktivasi pada setiap neuron. Jika suatu input telah melewati nilai *threshold* tertentu, maka kemudian neuron itu akan diaktifkan. Jika neuron tersebut telah diaktifkan, makan neuron tersebut akan mengirimkan output lewat bobot-bobot outputnya ke neuron yang lain yang berhubungan dengan neuron tersebut. Gambar 2.2 merupakan struktur dari Jarungan Syaraf Tiruan (Sinuhaji, 2009).



Gambar 2.2 Struktur Jaringan Syaraf Tiruan

Sumber : (Siwi, Cholissodin, & Furqon, 2016)

Jaringan syaraf tiruan dilatih agar input mengarah ke output yang spesifik. Jadi jaringan saraf tiruan ini akan dilatih terus sehingga mencapai titik dimana input sesuai dengan target yang ditentukan. Pelatihan dimana setiap input diasosiasikan dengan target yang telah ditentukan disebut pelatihan terarah (*Supervised learning*) (Fikriya, Irawan, & Soetrisno, 2017).

2.2.4 Fungsi Aktivasi

fungsi aktivasi adalah suatu operasi matematik yang berfungsi untuk mengaktifkan dan menonaktifkan neuron. Fungsi aktivasi ini digunakan dalam proses pelatian dan uji pada *Extreme Learning System*. Ada beberapa fungsi aktivasi yang dapat digunakan dalam Jaringan syaraf tiruan salah satunya adalah Fungsi aktivasi Sigmoid. Fungsi sigmoid itu sendiri dibagi menjadi dua, yaitu fungsi sigmoid biner dan fungsi sigmoid bipolar (Humaini, 2015).

1. Fungsi aktivasi Sigmoid Biner

Fungsi aktivasi sigmid biner ini memiliki nilai output sebesar 0 hingga 1. Fungsi aktivasi ini digunakan oleh jaringan syaraf tiruan yang memiliki outoput 0 sampai 1 atau 0 atau satu. Rumus persamaan fungsi aktivasi Sigmoid biner ditunjukkan pada Persamaan 2.1.

( ) ——— (2.1)

Keterangan:
( ) = Fungsi aktivasi *sigmoid*

= Eksponensial dengan pangkat minus data

dengan : ( ) ( ) ( )

2. Fungsi Aktivasi Sigmoid Bipolar

Fungsi Aktivasi Sigmoid bipolar ini hampir sama dengan Fungsi aktivasi sigmoid biner, yang membedakannya adalah outputnya saja, fungsi aktivai sigmoid biner ini memiliki keluaran range antara 1 hingga -1. Rumus dari fungsi sigmoid biner ditunjukkan pada Persamaan 2.2.

( ) ——— (2.2)

Keterangan:
( ) = Fungsi aktivasi *sigmoid*

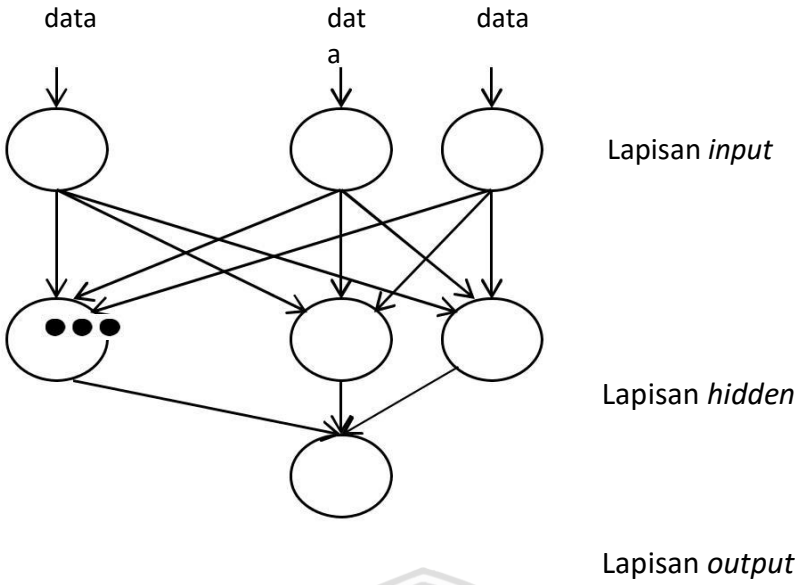
= Eksponensial dengan pangkat minus data

dengan : ( ) – ( ) ( )

2.2.5 Extreme Learning Machine (ELM)

Metode *Extreme Learning Machine* merupakan metode baru dari jaringan syaraf tiruan (JST) yang bersifat *feedforward* yang memiliki satu *hidden layer* atau biasa disebut dengan *single hidden layer feedforward neural network* (SLFNs). Metode ini dibuat untuk memperbaiki jaringan syaraf tiruan *feedforward* dalam hal kecepatan dalam belajar (*speed learning*). Algoritme *Extreme Learning Machine* ini tidak melatih bobot input maupun bias, tetapi *Extreme learning Machine* ini melatih untuk memperoleh bobot outputnya dengan menggunakan *norm-least-squares solution* dan *moore-penrose inverse* pada sistem linear secara umum. ELM dikatakan sebagai *learning machine* yang cepat karena metode ini mengginakan *input weight* dan bias secara random, sehingga ELM memiliki *learning speed* yang cepat (Humaini, 2015).

Perambatan maju atau *feedforward* adalah suatu jaringan lapis banyak (*multilayer network*). Jaringan ini menggunakan parameter-parameter yang telah ditentukan seperti *input weight* dan bias secara manual yang dibangkitkan secara acak dalam suatu rentang tertentu agar dapat menghindari hasil prediksi yang tidak stabil. Gambar 2.3 adalah ilustrasi dari struktur ELM (Humaini, 2015).



Gambar 2.3 Struktur Metode *Extreme Learning Machine*

Sumber : (Humaini, 2015)

Seperti yang digambarkan pada Gambar 2.3 diketahui bahwa prosesyang terjadi pada metode ini adalah data input akan diproses menggunakan *input weight* (w) dan bias (b) baik itu sebelum ataupun sesudah lapisan *hidden*, kemudian diproses dalam setiap neuron pada lapisan *hidden* menggunakan fungsi aktivasi sigmoid yang akhirnya menghasilkan hasil dari ELM pada lapisan *output*. Nilai bias dan *input weight* didapatkan secara random antara nilai 1 sampai -1. Langkah langkah perhitungan dengan menggunakan metode *Extreme Learning Machine* ini yang pertama-tama adalah normalisasi data, kemudian data yang telah ternormalisasi akan diproses pada proses *training* dan *testing* (Humaini, 2015).

2.2.6 Min Max Normalization

Normalisasi merupakan suatu transformasi data yang merupakan alat untuk *preprocessing* pada sistem *data mining*. Sebuah atribut dari data dinormalisasi dengan cara memproses *input* dengan nilai yang besar sehingga mengeluarkan *output* yang kecil sesuai dengan kebutuhan. Normalisasi sangat berguna dalam proses klasifikasi maupun yang menyakutpahutkan *neural network*, atau perhitungan jarak seperti *nearest neighbor classification and clustering.* Rumus dari *min max normalization* ditunjukkan pada Persamaan 2.3 (Jain & Bhandare, 2011).

_____ (2.3)

Keterangan:

= nilai dari hasil normalisasi data.

= nilai asli data.

min = niali min dari fitur Xn pada data. *max* = niali max dari fitur Xn pada data.

2.2.7 Proses Training

Proses ini adalah hal yang harus dilakukan sebelum melakukan pemerediksian. Tujian dari *training* itu sendiri adalah untuk memperoleh nilai *output weight.* Adapun langkah-langkah proses *Training menurut* Huang, Zhu dan Siew adalah sebagai berikut (Huang, Zhu, & Siew, 2006)

1. pertama, inisialisasi *input weight* dan bias. Nilai ini diinialisasi secara *random* dengan nilai antara 1 sampai -1.
2. Langkah ke dua, keluaran lapisan *hidden* dihitung menggunakan fungsi

   aktivasi. Langkah pertama adalah menghitung keluaran lapisan *hidden* ( ), setelah mendapatkan nilai , pada Persamaan 2.4 ditunjukkan persamaan untuk menghitung keluaran lapisan *hidden.*

   (Σ)

   (2.4)

   Keterangan:

   = Matriks keluaran *hidden layer*.

   = *1,2,….,*N*], dimana *N* adalah keseluruhan jumlah data.

   = *1,2,…., *Ñ*], dimana *Ñ* adalah keseluruhan jumlah *hidden neuron*.

   = Jumlah *input neuron*.

   = *input weight*.

   = *Input* data yang digunakan.

   = Nilai bias

3. tahap ketiga adalah mencari matriks keluaran *hidden layer* hitung hasil dari tersebut dengan fungsi aktivasi sigmoid (H(x)) seperti pada Persamaan 2.2.
4. tahap ke empat adalah melakukan t*ranspose* matriks keluaran *hidden layer* dengan fungsi aktivasi sigmoid ( ).
5. tahap kelima adalah mengkalikan keluaran *hidden layer* dengan fungsi aktivasi sigmoid ( ) dengan matriks keluaran *hidden layer* dengan fungsi aktivasi sigmoid (H(x)).
6. setelah tahap ke-5 sudah terselesaikan, maka matriks dari perkalian tersebut diinverskan, dan kemudian hasil matriks invers tersebut dikalikan dengan matriks keluaran *hidden layer* dengan fungsi aktivasi sigmoid yang telah ditranspose. Keluaran dari perkalian tersebut disebut dengan *Moore-Penrose Generalized Invers* ( ) sesuai dengan Persamaan 2.5.

   =( )

   (2.5)

7. langkah ke tujuh adalah menhitung *output weight.* Langkah langkah dalam menghitung *Output weight* adalah sesuai dengan Persamaan 2.6**.**

(2.6)

Keterangan:

= Matriks *Output weight*.

= Matriks *Moore-Penrose Generalized Invers* dari matriks .

= Matriks Target

2.2.8 Testing

Proses ini berfungsi untuk mendapatkan hasil prediksi dengan mengevaluasi *Extreme Learning Machine* dari proses *training* sebelumnya. Proses ini menggunakan output*, input weight*, dan bias yang dihasilkan pada proses *training* sebelumnya. Adapun langkah-langkah proses *testing menurut* Huang, Zhu dan Siew adalah sebagai berikut (Huang, Zhu, & Siew, 2006).

1. pertama yang harus dilakukan adalah inialisasi *input weight* dan bias yang diperoleh dari proses *training.*
2. Keluaran pada lapisan *hidden* dihitung menggunakan fungsi aktivasi sigmoid.
3. Langkah berikutnya adalah menghitung nilai *output layer* dari *output weight* yang didapatkan dari proses *training. Output layer* adalah hasil prediksi. Pada Persamaan 2.7 ditunjukkan persamaan untuk mengitung nilai *output layer.*

   (2.7)

   = *Output layer* yang merupakan hasil prediksi.

   = nilai *output weight* didapatkan dari proses *training*.

   = Keluaran di *hidden layer* dihitung dengan fungsi aktivasi.

4. langkah yang terakhir adalah menghitung nlai error semua output layer dimana nilai eror ini adalah nilai kesalahan prediksi. Untuk menghitung nilai eror ini digunakan perhitungan *Mean Square Error*(MSE) yang akan ditunjukkan pada Persamaan 2.9**.**

2.2.9 Proses Denormalisasi Data

Proses ini adalah kebalikan dari normalisasi dimana data dinormalisasi dengan cara memproses input dengan nilai yang besar sehingga mengeluarkan output yang kecil sesuai dengan kebutuhan, sedangkan denormalisasi adalah kebalikannya yaitu mengembalikan nilai yang diperkecil pada normalisasi menjadi nilai asli (Triyono, 2011)

pada Persamaan 2.8 ditunjukkan persamaan untuk melakukan denormalisasi (Perdana, 2016).

( ) (2.8)

Keterangan:

= nilai hasil prediksi sebelum didenormalisasi

= nilai asli setelah didenormalisasi

min = nilai minimum pada *data* fitur *x*

max = nilai maksimal pada *data* fitur *x*

2.2.10 Mean Square Error (MSE)

Mean Square Error adalah salahsatu jenis perhitungan yang digunakan untuk mengevaluasi berapa tingkat error pada suatu proses prediksi. Pada Persamaan 2.9 akan ditunjukkan perhitungan *mean square eror* (MSE) (Triyono, 2011).

$$\frac{\Sigma}{} \quad \frac{\Sigma(\quad)}{} \tag{2.9}$$

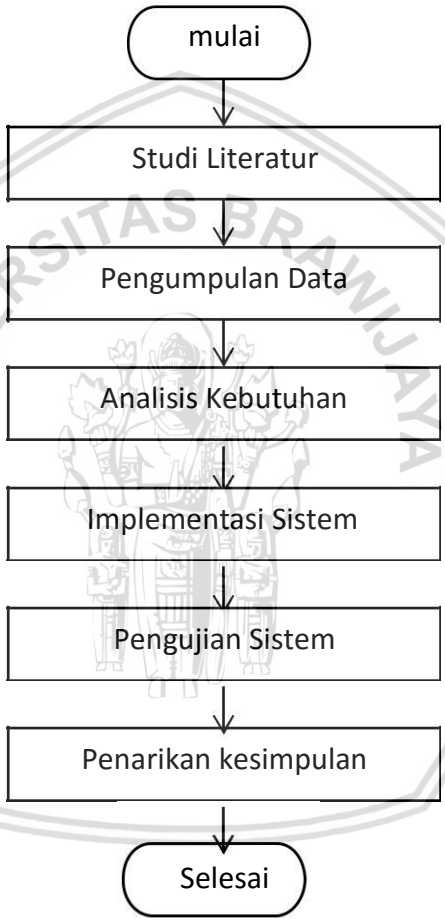
Keterangan:

= Jumlah data = *Error.*

= Nilai *output* (prediksi). = Nilai aktual

BAB 3 METODOLOGI

Pada bab ketiga ini, berisi tentang penjelasan metode dan langkah-langkah yang digunakan untuk menyusun skripsi ini. Langkah-langkah yang digunakan untuk menyusun skripsi ini yaitu meliputi studi literatur, analisis kebutuhan sistem, pengumpulan data, perancangan sistem, implementasi sistem, pengujian sistem dan yang terakhir adalah penarikan kesimpulan. Untuk lebih jelasnya bisa dilihat pada *flowchart* Metodologi Penelitian yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Studi literatur adalah hal yang pertama dilakukan oleh peneliti yaitu proses mempelajari tentang penunjang dasar teori yang diterapkan atau digunakan pada penelitian ini serta penelitian-penelitian sebelumnya yang berhubungan dengan skripsi ini. Sumber yang peneliti gunakan untuk studi literatur ini yakni : jurnal nasional maupun internasional, buku, *e-book,* halaman web, dan skripsi atau laporan penelitian sebelumnya yang menunjang skripsi ini. Teori yang

berkaitan dengan Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*), diantaranya adalah : *Bakery*, prediksi, Jaringan Syaraf Tiruan, metode *Extreme Learning Machine*(ELM), metode untuk menormalisasi data yaitu metode *Min-Max Normalization*, dan perhitungan *Mean Square Error*(MSE) untuk mencari tingkat error pada aplikasi ini.

3.2 Pengumpulan Data

Pengumpulan data pada penelitian ini diambil data penjualan Harum *Bakery.* Pengambilan data ini dilakukan melalui proses izin terhadap *Owner* dari Harum *Bakery* itu sendiri untuk meminta data. Data yang diambil adalah data penjualan roti pada perusahaan Harum *Bakery* dalam kurun waktu 5 bulan terakhir. Mulai dari bulan September 2017 hingga Januari 2018. Dengan 3 jenis roti, yakni roti tawar, manis, dan cake.

3.3 Analisis kebutuhan

Pada analisis kebutuhan ini, berfungsi untuk mengetahui dan menganalisis kebutuhan apa saja yang dibutuhkan dalam Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*) dimulai dari analisis kenbutuhan perangkat keras dan perangkat lunak yang akan digunakan dalam sistem.

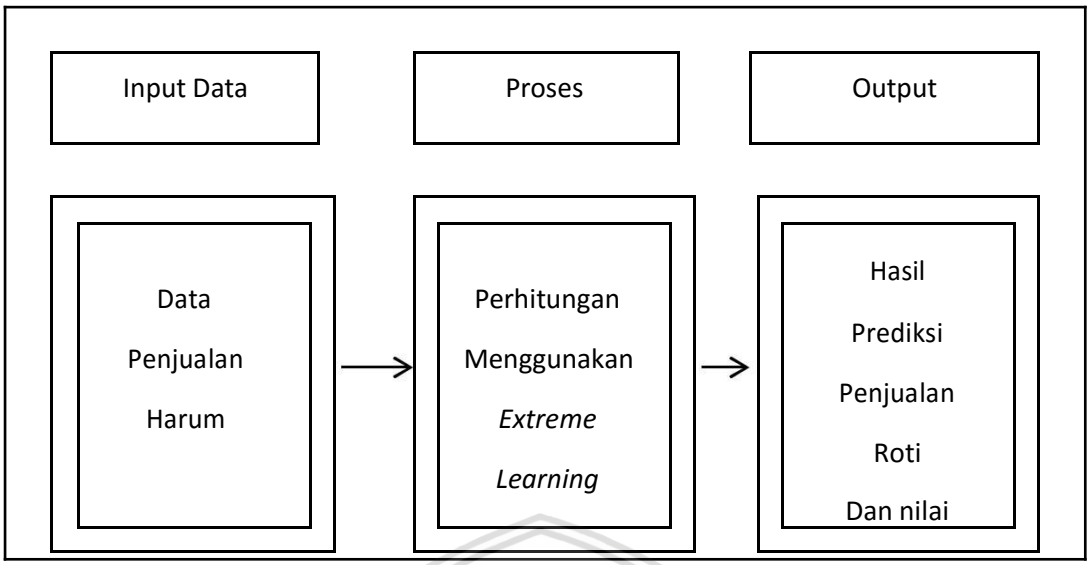
1. Spesifikasi kebutuhan *Hardware Android Device*:

- CPU : 1,3 GHz
- RAM : 2GB
- Storage : 100 MB
- Android 4.4.4 KitKat

2. Spesifikasi kebutuhan *Software* :

- DBMS : Firebase database
- Development tools : Android Studio
- Word Processoe : MS word, MS Excel 2016
- Bahasa Pemrograman : Java
- OS : Windows 10 64bit

Sistem ini menggunakan masukkan yaitu data penjualan Harum Bakery, jumlah *neuron* pada *hidden layer,* dan jumlah variasi fitur, kemudian data diproses menggunakan metode ELM, dengan output berupa prediksi penjualan roti dan tingkat kesalahan *Mean Square Error*(MSE). Gambar 3.2. menununjukkan arsitektur dari sistem aplikasi prediksi penjualan roti.



Gambar 3.2 Diagram Blok Perancangan Sistem

3.4 Implementasi Sistem

Pada tahap implementasi sistem ini dilakukan implementasi dari tahapan perancangan sistem sebelumnya. Penerapan metode *Extreme Learning Machine* untuk prediksi penjualan roti pada Harum *Bakery* ini merupakan aplikasi berbasis *mobile/Android*. Implementasi yang dilakukan yaitu implementasi *interface*, *database*, dan algoritme/metode. Implementasi dari perancangan ini dilakukan dengan menggunakan bahasa pemrograman Java dengan menggunakan metode *Extreme Learning Machine.* Pada tahapan ini juga akan dipaparkan manualisasi perhitungan sederhana dari skripsi yang penulis teliti.

3.5 Pengujian Sistem

pada pengujian system ini, berfungsi untuk menguji sistem menggunakan metode *Extreme Learning Machine* sebagai metode prediksi penjualan roti di Harum *Bakery* yang kemudian data akan diuji dengan bebrapa kali pengujian, dengan berbagai parameter pengujian dan nantinya akan diukur nilai tingkat erornya menggunakan metode *Mean Square Error*(MSE) yang mana peneliti melakukan pengujian tersebut untuk mencari nilai MSE yang terkecil (memiliki nilai error terkecil).

3.6 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah semua proses telah dilakukan mulai dari perancangan hingga pengujian sistem Implementasi Metode *Extreme Learning Machine* (ELM) untuk memprediksikan penjualan roti (Studi Kasus : Harum *Bakery*). Kesimpulan diambil dari pengujian dan analisis metode yang diterapkan serta juga merupakan jawaban dari rumusan masalah yang telah dipaparkan sebelumnya pada bab Pendahuluan. Pada tahap yang terakhir adalah

saran. Pada saran ini berisi tentang saran dari penulis tentang hasil yang telah dicapai dan untuk pengembang berikutnya untuk memperbaiki kelemahan yang ada pada penelitian ini.

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Permasalahan yang dihadapi dan akan diselesaikan adalah memprediksi penjualan di sebuah toko roti/*bakery* tepatnya peneliti menggunakan Harum *Bakery* sebagai studi kasus yang menggunakan metode *Extreme Learning Machine* (ELM) untuk membantu perusahaan dalam prediksian penjualan roti. *Input* untuk prediksi penjualan roti menggunakan metode *Extreme Learning Machine* ini adalah data historis penjualan roti manis, roti tawar, dan roti cake perhari, variasi fitur data historis penjualan, dan jumah *neuron* pada *hidden layer.* Parameter perhitungan pada skripsi ini meliputi: jumlah data, fitur atau variasi yang digunakan, fungsi aktivasi dan jumlah neuron pada *hidden layer*. Hal yang dilakukan setelah melakukan *collect data* penjualan roti pada Harum *Bakery* adalah membuat Data *training* dan data *testing* yang data tersebut harus dinormalisasi menggunakan *Min Max Normalization* sebelum maju ke tahap berikutnya, kemudian ada proses *training data* dan *testing data* yang kemudian dihitung nilai eror prediksi mengunakan *Mean Square Error* (MSE), dan kemudian data prediksi tersebut akan didenormalisasi kembali. Nilai dari keluaran tersebut dibandingkan dengan nilai asli penjualan roti dan kemudin diproses menggunakan MSE. Keluaran dari aplikasi ini adalah nilai MSE dan nilai prediksi yang telah didenormalisasi. Data sampel historis data penjualan untuk produk roti tawar, roti manis, dan roti cake perhari ditunjukkan pada Tabel 4.1.

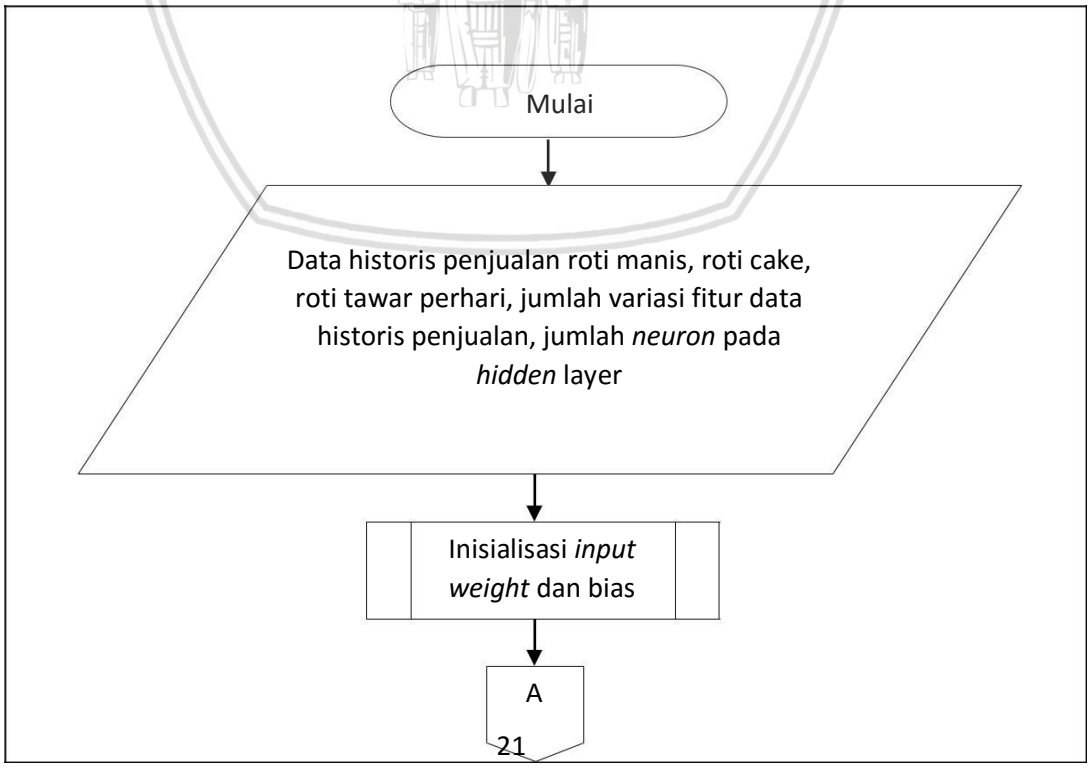
Tabel 4.1 Data Sampel Historis Penjualan Harian

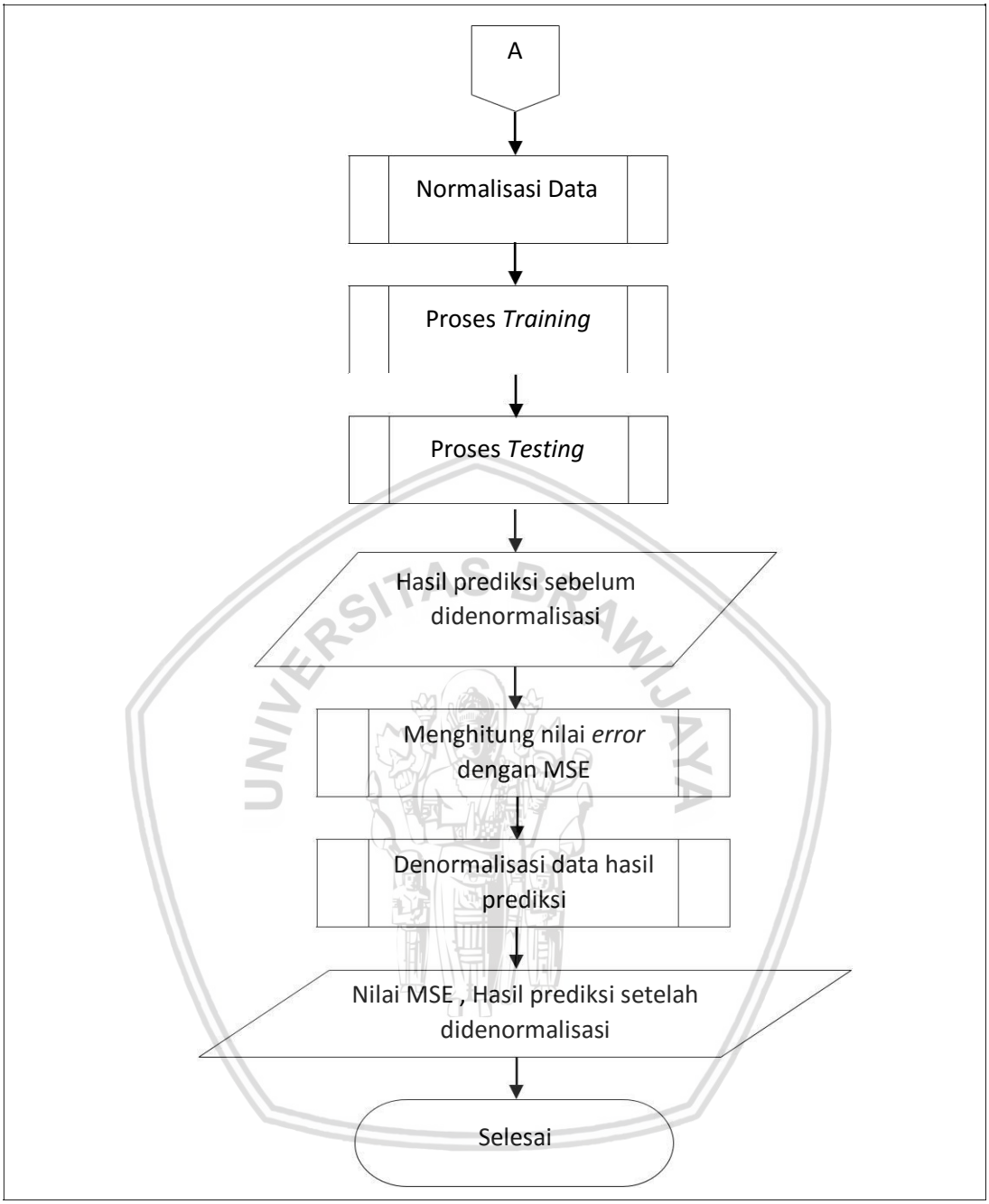
tanggal	Jenis roti		
	Roti Manis	Roti Tawar	Roti Cake
1/12/2017	257	15	5
2/12/2017	101	25	4
3/12/2017	136	13	10
4/12/2017	136	9	17
5/12/2017	121	14	11
6/12/2017	165	12	19
7/12/2017	139	11	12
8/12/2017	154	15	16
9/12/2017	113	15	16
10/12/2017	224	13	4

4.2Penyelesaian Permasalahan Prediksi Penjualan Roti Menggunakan Metode *Extreme Learning Machine* (ELM)

Prediksi penjualan roti ini adalah untuk mengetahui berapa tingkat eror yang dihasilkan dari proses MSE jika prediksian ini menggunakan metode ELM. Data Input untuk prediksi penjualan roti ini berupa data historis penjualan roti manis, roti tawar, dan roti cake perhari. Penelitian ini memiliki beberapa *steps* dalam mendapatkan nilai hasil prediksi penjualan roti yang kemudian langkah-langkahnya ditunjukkan pada Gambar 4.1 yang merupakan diagram alir untuk proses prediksi penjualan menggunakan metode *Extreme Learning Machine* dari mulai dari proses *input* hingga proses *output*. Pada penelitian ini, peneliti menggunakan metode *Min-Max Normalization* sebagai metode penormalisasian data dan menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya.

Pada Gambar 4.1 merupakan *flowchart* proses prediksi penjualan roti menggunakan metode *Extreme Learning Machine* pada Harum *Bakery* yang menjelaskan alur dari proses pola masukkan, kemudian inisialisasi *input weight* dan bias, sebelum data dilakukan proses *training* dan *testing,* data harus di normalisasi terlebih dahulu karena data yang diolah merupakan data dalam *range* yang berbeda, kemudian setelah melalui *testing* akan menghasilkan prediksi dan kemudian hasil prediksi tersebut didenormalisasi sehingga keluaran hasil prediksi dapat dikonversikan lagi ke angka sesuai dengan data aslinya. Dan sebelum data dikonversikan lagi ke bilangan asli, dari hasil proses *testing* tersebut, peneliti dapat menentukan nilai tingkat error dalam prediksiannya menggunakan metode MSE. Gambar 4.1 merupakan *flowchart* proses prediksi penjualan roti menggunakan metode ELM





Gambar 4.1 *Flowchart* Proses Prediksi Menggunakan *Extreme Learning Machine*

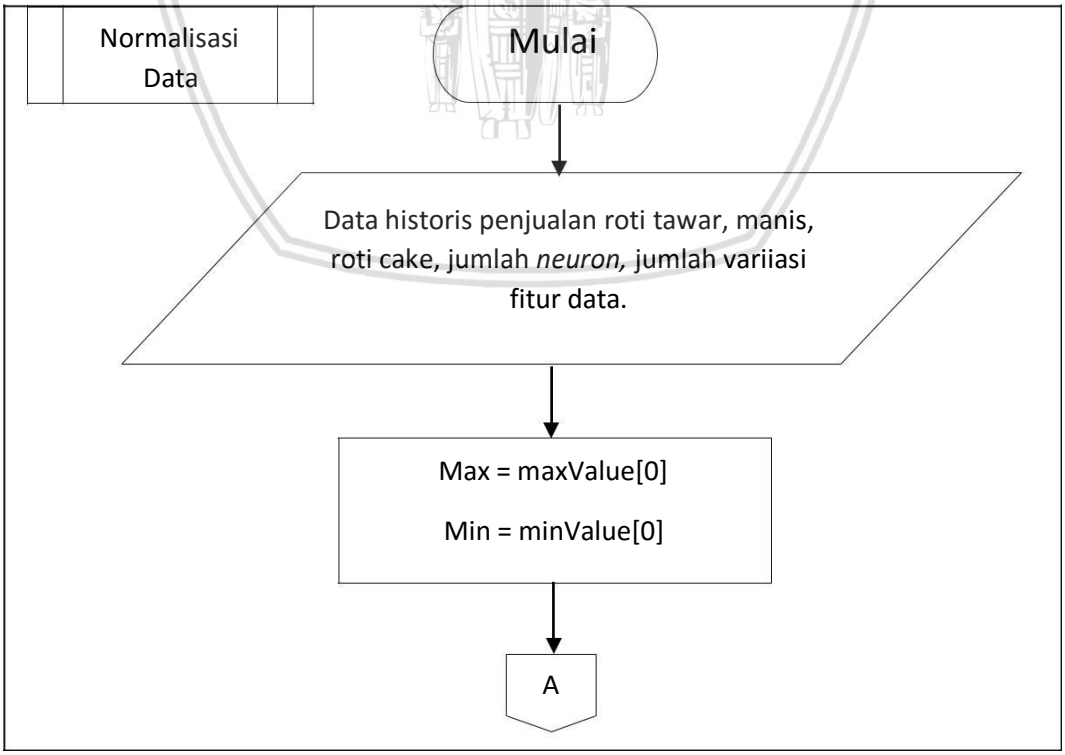
Berdasarkan *Flowchart* proses prediksi pada Gambar 4.3, dijelaskan langkah-langkah untuk menyelesaikan masalah prediksi dengan menggunakan metode *Extreme Learning Machine*:

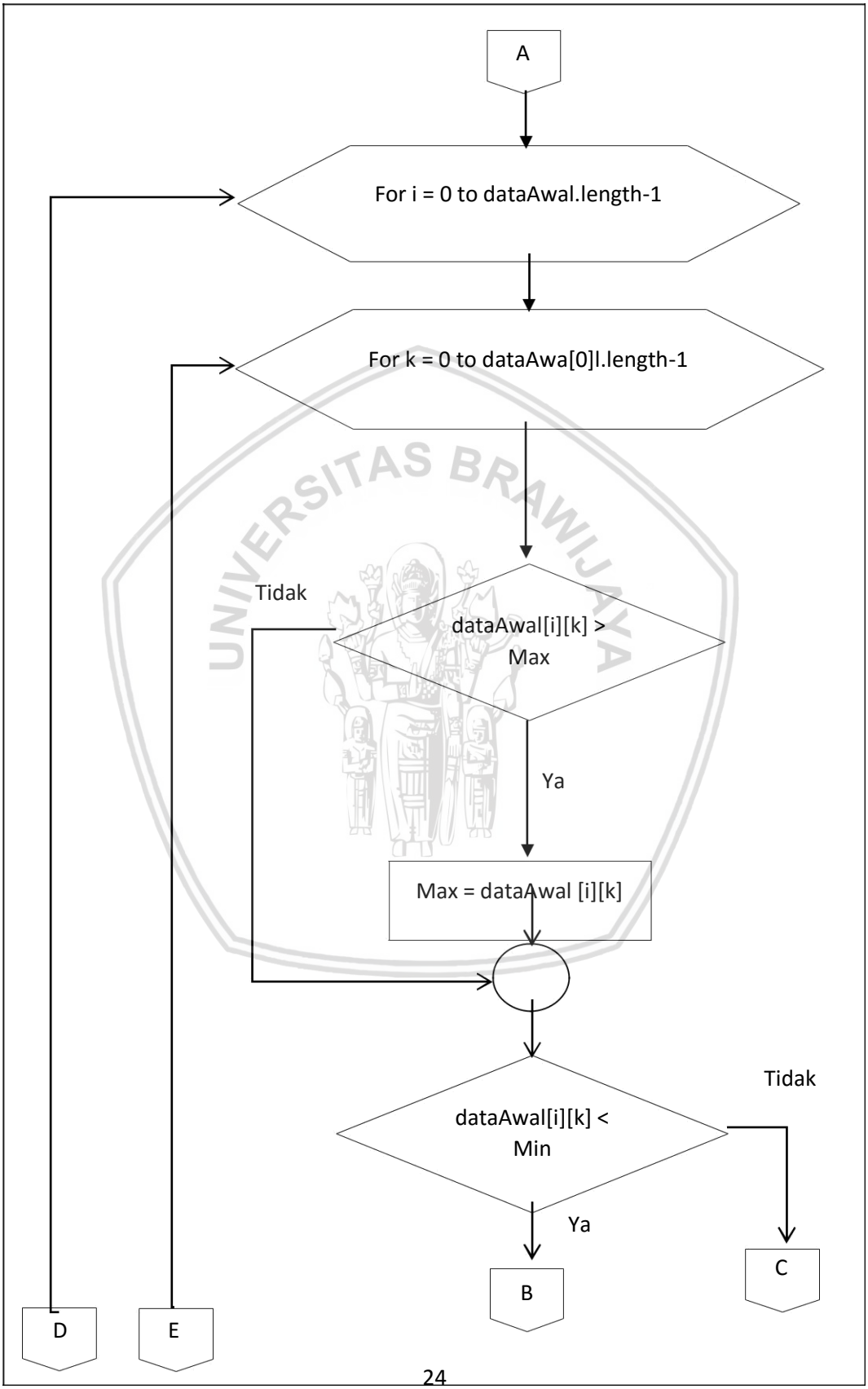
1. Sistem menerima masukkan berupa data historis penjualan perhari, jumlah *neuron* pada *hidden layer*, dan jumlah fitur. Penelitian ini menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya.

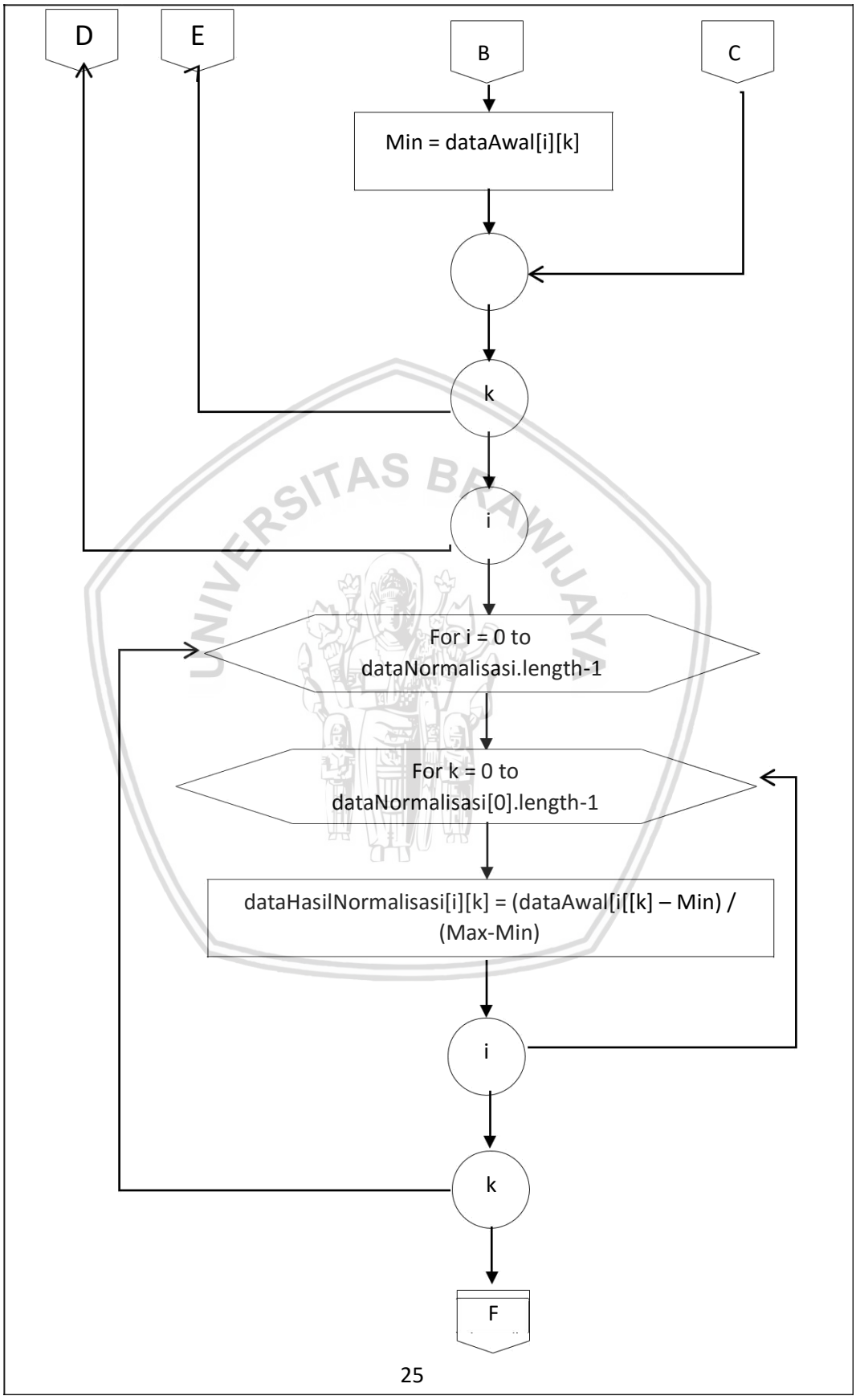
2. Inisialisasi *input weight* dan bias yang flowchartnya ditunjukkan pada Gambar 4.4.
3. Proses normalisasi data pada penelitian ini menggunakan normalisasi *min max normalization* yang kemudian diagram alirnya ditunjukkan pada Gambar 4.5.
4. Data yang telah dinormalisasi kemudian dilakukan proses *training* yang diagram alirnya ditunjukkan pada Gambar 4.6 dan setelah itu data akan dilakukan proses *testing* yang flowchartnya ditunjukkan pada Gambar 4.9. Hasil dari proses *testing* adalah sebuah prediksi namun masih belum dapat dibaca karena belum didenormalisasi, maka dari itu hasil dari proses *testing* tersebut harus didenormalisasi terlebih dahulu.
5. Hasil prediksi setelah proses *testing* dan sebelum didenormalisasi tersebut kemudian dievaluasi dengan menghitung nilai erornya menggunakan *Mean Square Eror* (MSE) yang persamaannya ditunjukkan pada Persamaan 2.8. Diagram alir dari perhitungan nilai *Mean Square Error* ditunjukkan pada Gambar 4.9.
6. Hasil *output* dari sistem adalah hasil prediksi yang telah melalui tahap didenormalisasi.

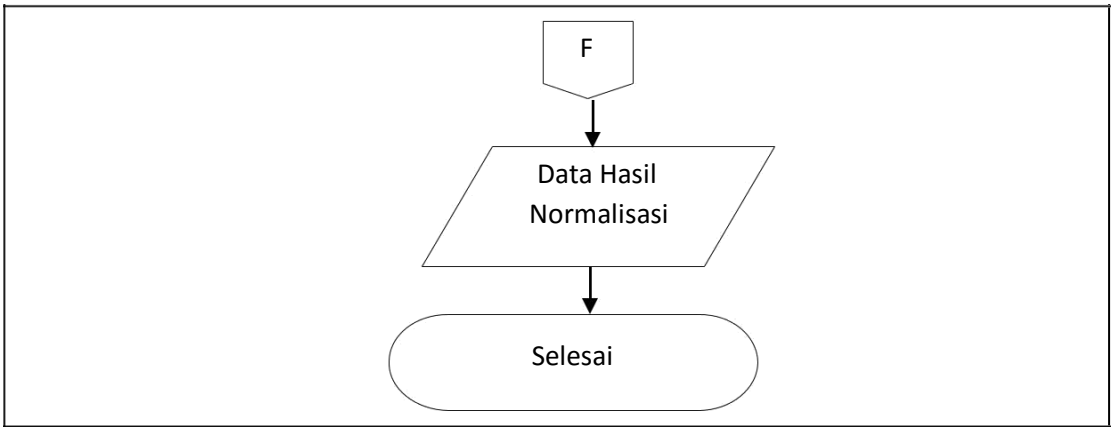
4.2.1 Proses Normalisasi Data

Proses normalisasi data ini berfungsi untuk merubah bilangan asli ke bilangan desimal yang bertujuan untuk menyamaratakan angka dalam perhitungan pada metode ELM ini. *Range* nilai pada normalisasi ini adalah dari 0 sampai 1. Pada penelitian ini, peneliti menggunakan metode *Min-Max Normalization.* Gambar 4.2 adalah *flowchart* dari proses normalisasi data.




A
For i = 0 to dataAwal.length-1
For k = 0 to dataAwa[0]l.length-1
Tidak
dataAwal[i][k] > Max
Ya
Max = dataAwal [i][k]
Tidak
dataAwal[i][k] < Min
Ya
B
C
D
E


D
E
B
C
Min = dataAwal[i][k]
k
i
For i = 0 to
dataNormalisasi.length-1
For k = 0 to
dataNormalisasi[0].length-1
dataHasilNormalisasi[i][k] = (dataAwal[i[[k] – Min) /
(Max-Min)
i
k
F

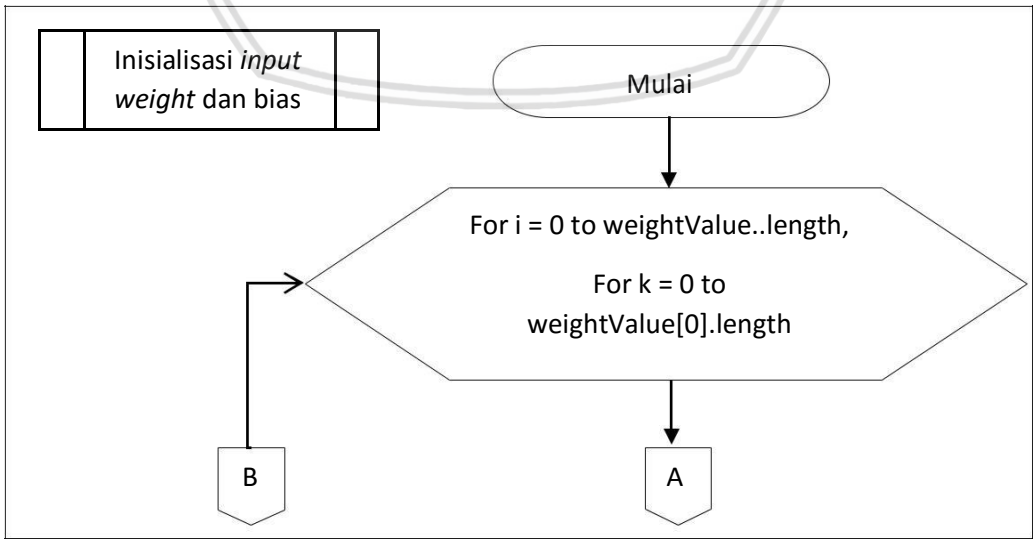


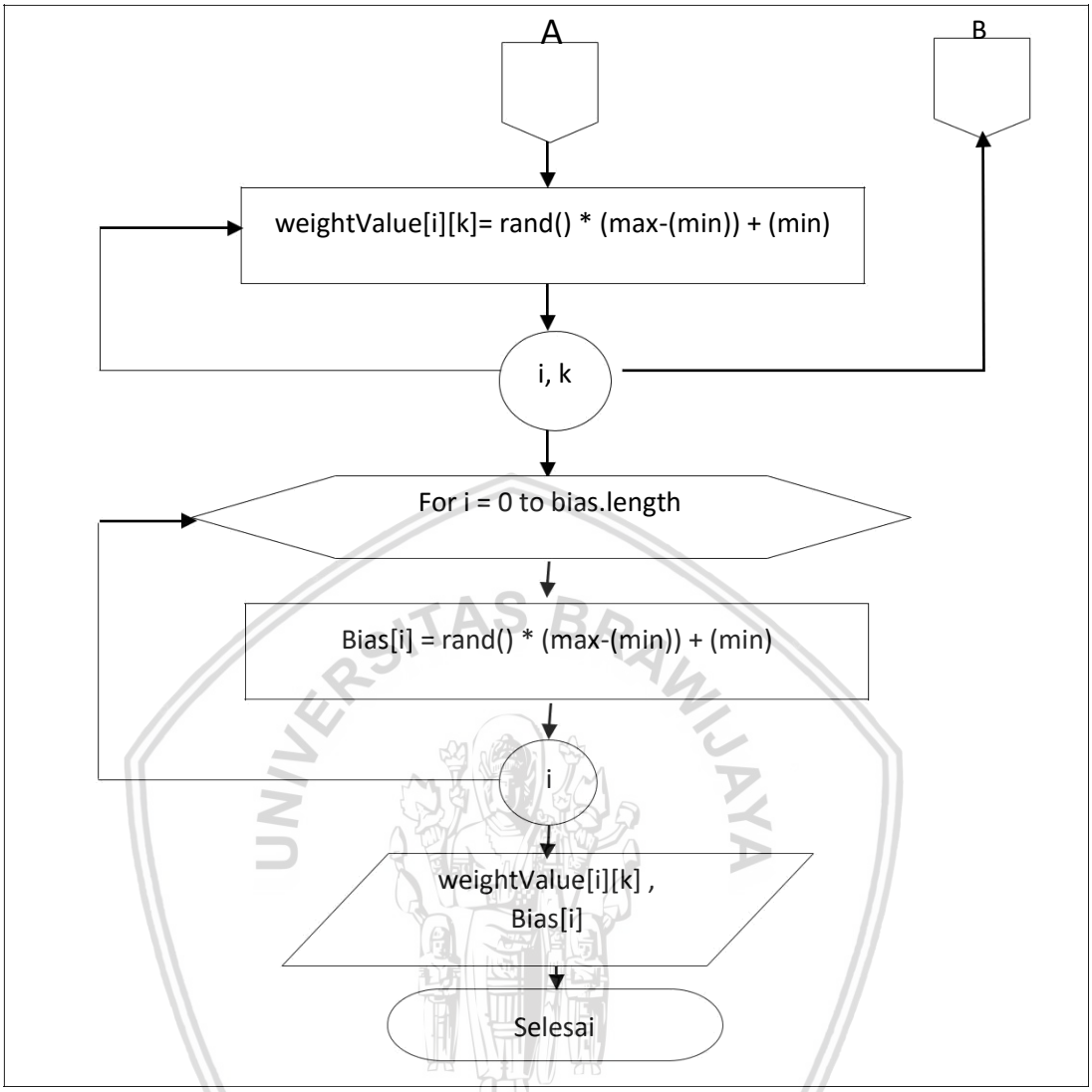
Gambar 4.2 *Flowchart* Proses Normalisasi

Flowchart pada Gambar 4.2 ditunjukkan *steps* normalisasi data menggunakan metode *Min-Max Normalization.* Langkah awal yang dilakukan adalah menerima data masukkan berupa data penjualan roti manis, roti cake, dan roti tawar. Kemudian setelah data diinputkan, maka sistem akan mencari nilai maksimal (Max) dan nilai minimal (Min). Setelah didapatkan nilai min dan max nya, maka data akan dihitung menggunakan Persamaan 2.3. seiap data masukkan akan diolah menggunakan Persamaan 2.3 dan kemudian data tersebut menjadi data yang sudah ternormalisasi.

4.2.2 Proses Inisialisasi *Input Weight* dan Bias

Diproses inisialisasi *input weight dan bias* ini, nilainya diinputkan secara rancom dengan nilai rentang antara -1 sampai dengan 1. Jumlah dari *weight* dan bias ini disesuaikan dengan masukkan dari jumlah neuron pada *hidden layer*. Gambar 4.3 merupakan *flowchart* dari proses inisalisasi *input weight* dan bias.





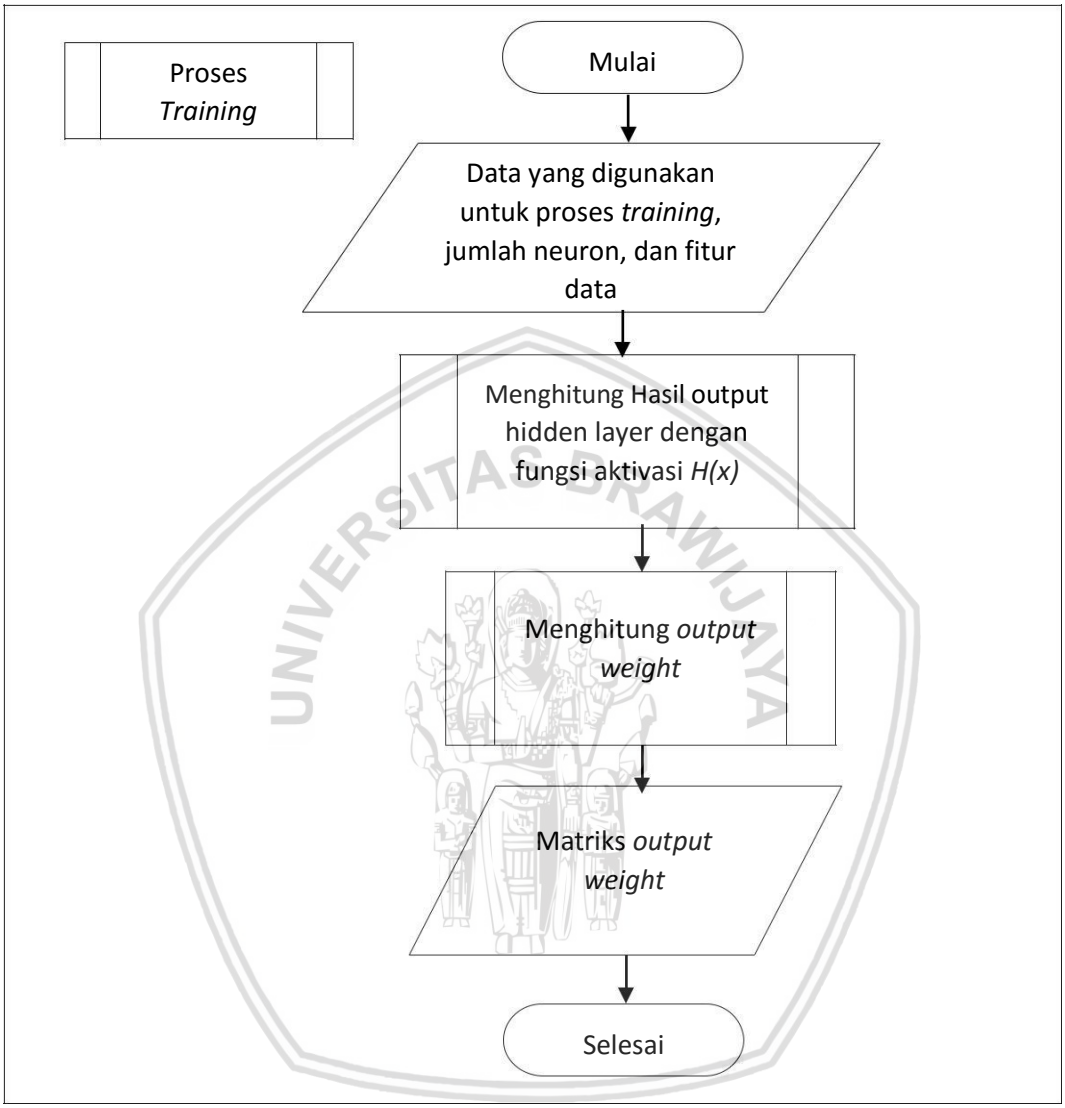
Gambar 4.3 *Flowchart* Proses Inisialisasi *Input Weight* dan *Bias*

Flowchart pada Gambar 4.3 ditunjukkan *steps* dari proses inisialisasi *input weight* dan bias. Hal pertama yang dilakukan adalah melakukan perulangan sebanyak jumlah *hidden neuron* dan jumlah input *neuron.* Proses yang selanjutnya adalah membangkitkan nilai *input weight* dan bias ini dilakukan secara *random* dengan rentang nilai -1 sampai 1.

4.2.3 Proses Training

Dengan menggunakan *Input weight* dan *bias* yang telah didapatkan pada proses inisialisasi sebelumnya, proses *training* disini akan menggunakan *Input weight* dan *bias* tersebut untuk menghasilkan nilai *output weight*. Untuk menghitung *output weight* diperlukan nilai *output hidden layer.* Setelah nilai *output hidden layer* didapatkan, nilai *output hidden layer* diproses lagi

menggunakan fungsi aktivasi sigmoid. Nilai output dari proses ini kemudian akan digunakan sebagai data untuk proses *Testing.* Gambar 4.4 merupakan *flowchart* proses *training* yang mana didalamnya terdapat langkah-langkah untuk proses *Training*.



Gambar 4.4 *Flowchart* Proses *Training*

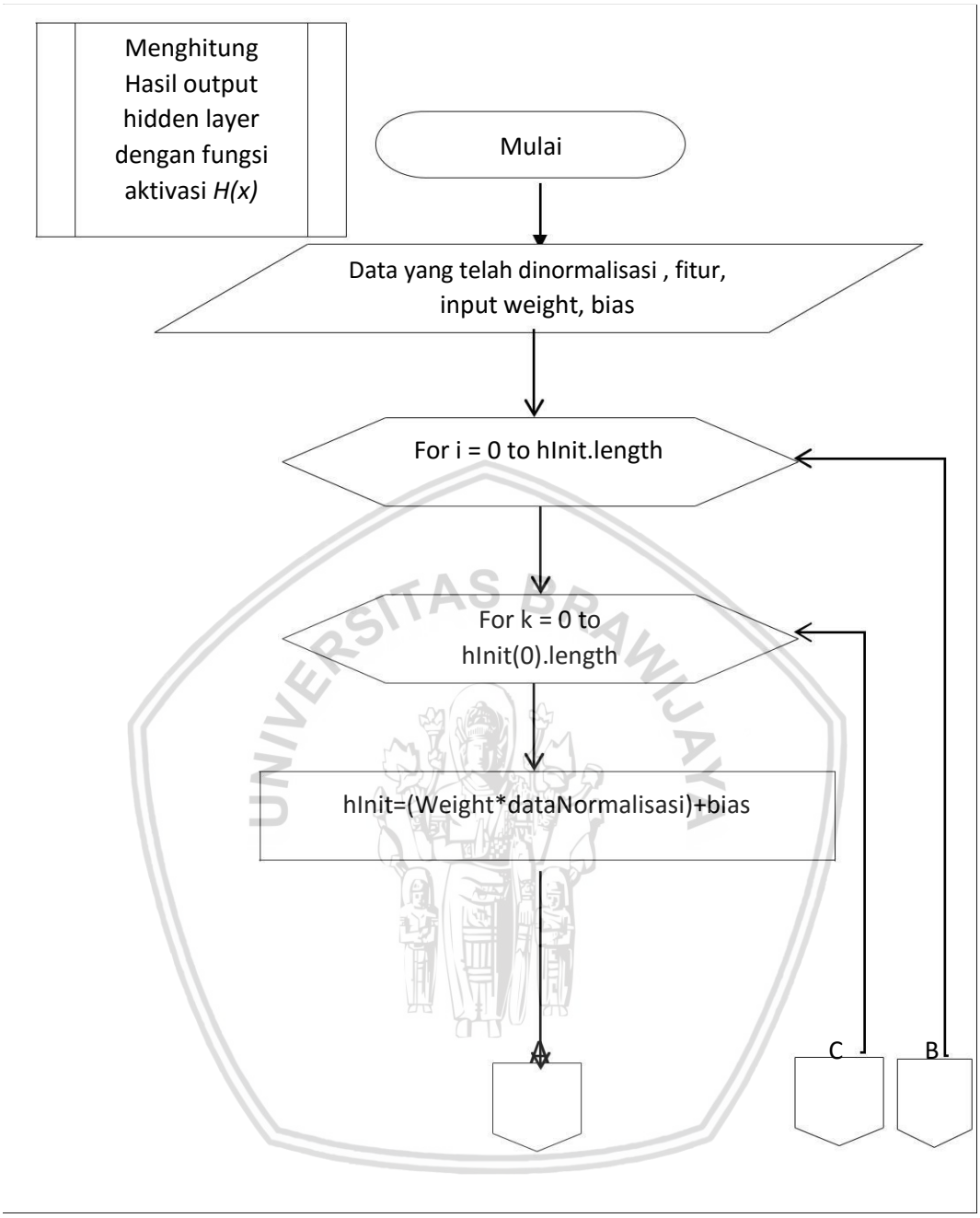
Berdasarkan *Flowchart* proses *Training* pada Gambar 4.6, dijelaskan langkah-langkah untuk menyelesaikan proses *Training* dimana langakah-langkahnya sebagai berikut :

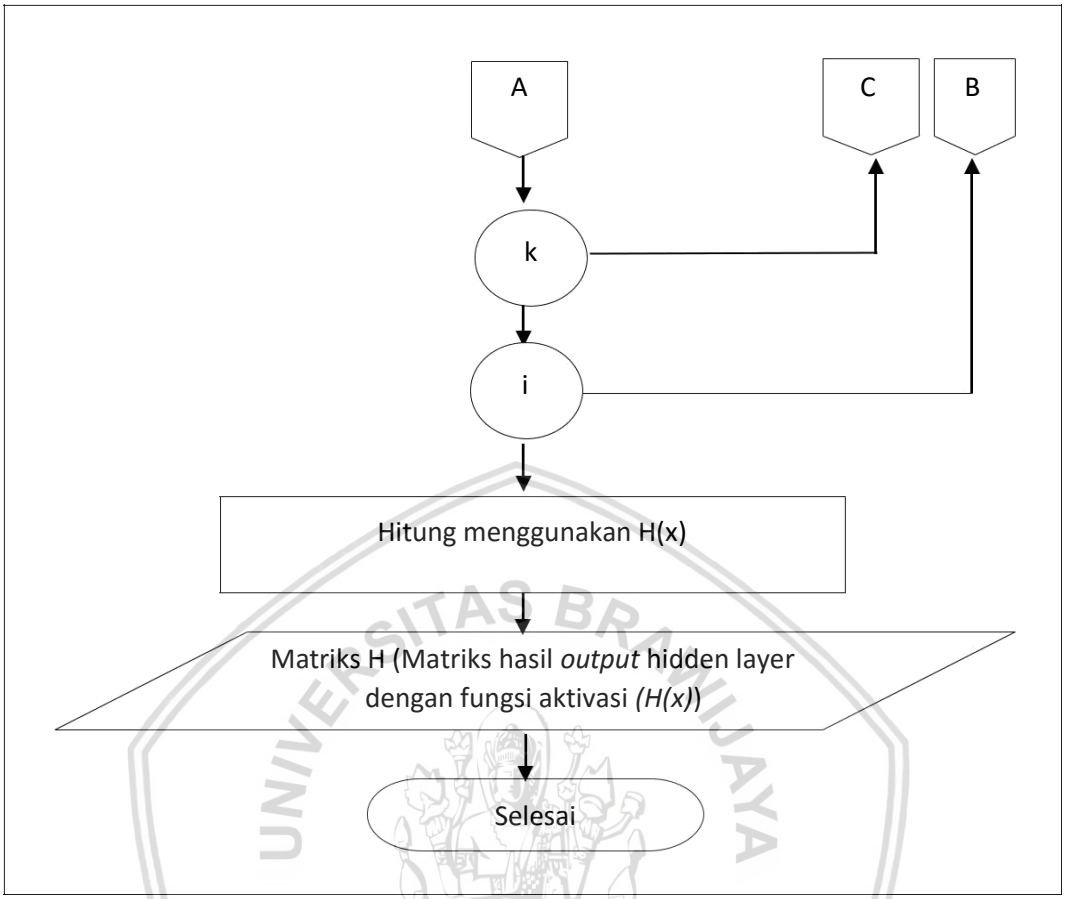
1. Sistem menerima masukkan yaitu data yang digunakan sebagai data training, jumlah neuron, dan pada penelitian ini peneliti menggunakan fungsi aktivasi seigmoid sebagai fungsi aktivasinya.
2. *Output hidden layer* dihitung menggunakan fungsi aktivasi sigmoid yang *flowchart* langkah-langkah untuk menyelesaikannya ditunjukkan pada Gambar 4.5.

3. Proses perhitungan *output weight* ini ditunjukkan pada *flowchart* menghitung *output weight* pada Gambar 4.6.
4. Nilai output dari sistem adalah berupa matriks *output weight*.

4.2.3.1 Proses Menghitung Hasil Output Hidden Layer menggunakan Fungsi Aktivasi Sigmoid H(x)

Tahap ini merupakan proses untuk menghitung hasil dari *output hidden layer* kemudian dihitung dengan menggunakan fungsi aktivasi yang mana pada penelitian ini peneliti menggunakan fungsi aktivasi sigmoid. Hal pertama yang dilakukan yaitu menghitung *output hidden* layer kemudian hasil perhitungan tersebut diproses lagi menggunakan fungsi aktivasi sigmoid. Gambar 4.5 merupakan *flowchart* dari proses menghitung hasil *output hidden layer*


Menghitung Hasil output hidden layer dengan fungsi aktivasi *H(x)*
Mulai
Data yang telah dinormalisasi , fitur, input weight, bias
For i = 0 to hInit.length
For k = 0 to hInit(0).length
hInit=(Weight*dataNormalisasi)+bias
A
C
B



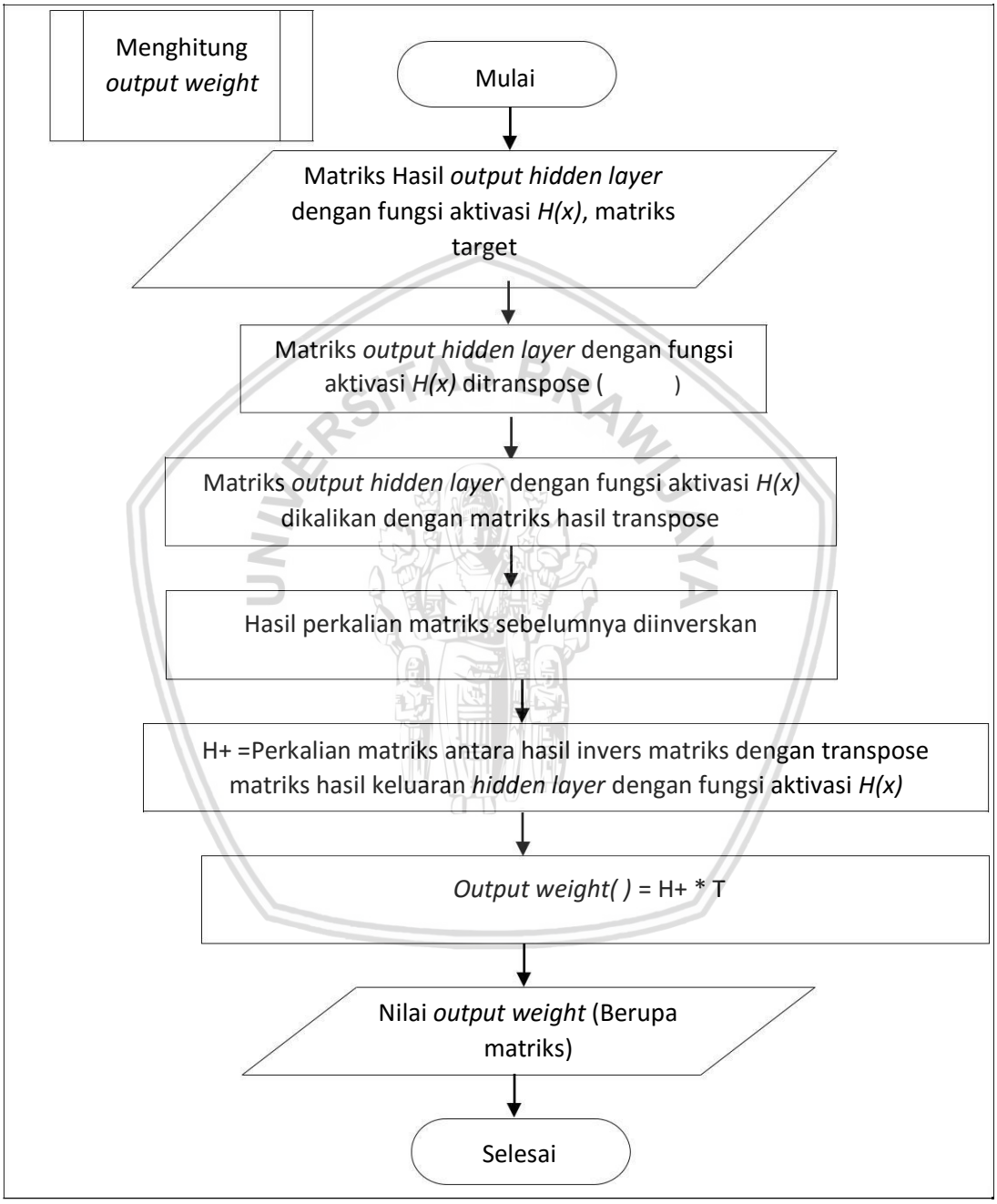
Gambar 4.5 *Flowchart* Proses Menghitung *Output Hidden Layer* dengan Fungsi Aktivasi Sigmoid *(H(x))*

Berdasarkan *Flowchart* proses *Training* pada Gambar 4.5. dijelaskan langkah-langkah untuk menghitung *output hidden layer* menggunakan fungsi aktivasi sigmoid dimana langakah-langkahnya sebagai berikut :

1. Data yang sudah dinormalisasi pada langkah awal kemudian dipakai pada proses perhitungan *output hidden layer* dengan fungsi aktivasi sigmoid. Data tersebut kemudian digunakan sesuai dengan proses yang dijalankan, sebagai contoh, jika proses yang sedang dijalankan adalah proses *training* maka data yang digunakan adalah data untuk *training* begitu pula sebaliknya pada proses *testing.*
2. Data *training* yang ternormalisasi tersebut kemudian dihitung *output hidden layer* secara keseluruhan data *training* sesuai dengan persamaan 2.4.
3. Setelah data yang ternormalisasi tersebut sudah dihitung *output hidden* layernya, maka hasilnya dihitung dengan fungsi aktivasi sigmoid *H(x)*.
4. Output dari perhitungan diatas akan menghasilkan matriks *output hidden layer* dengan fungsi aktivasi sigmoid.

4.2.3.2 Proses Menghitung Output Weight

Pada tahap ini, merupakan suatu proses yang bertujuan untuk mendapatkan nilai matriks *output weight* yang kemudian matriks *output weight* ini digunakan dalam proses *testing*. Gambar 4.6 merupakan *flowchart* dari proses menghitung *output weight.*



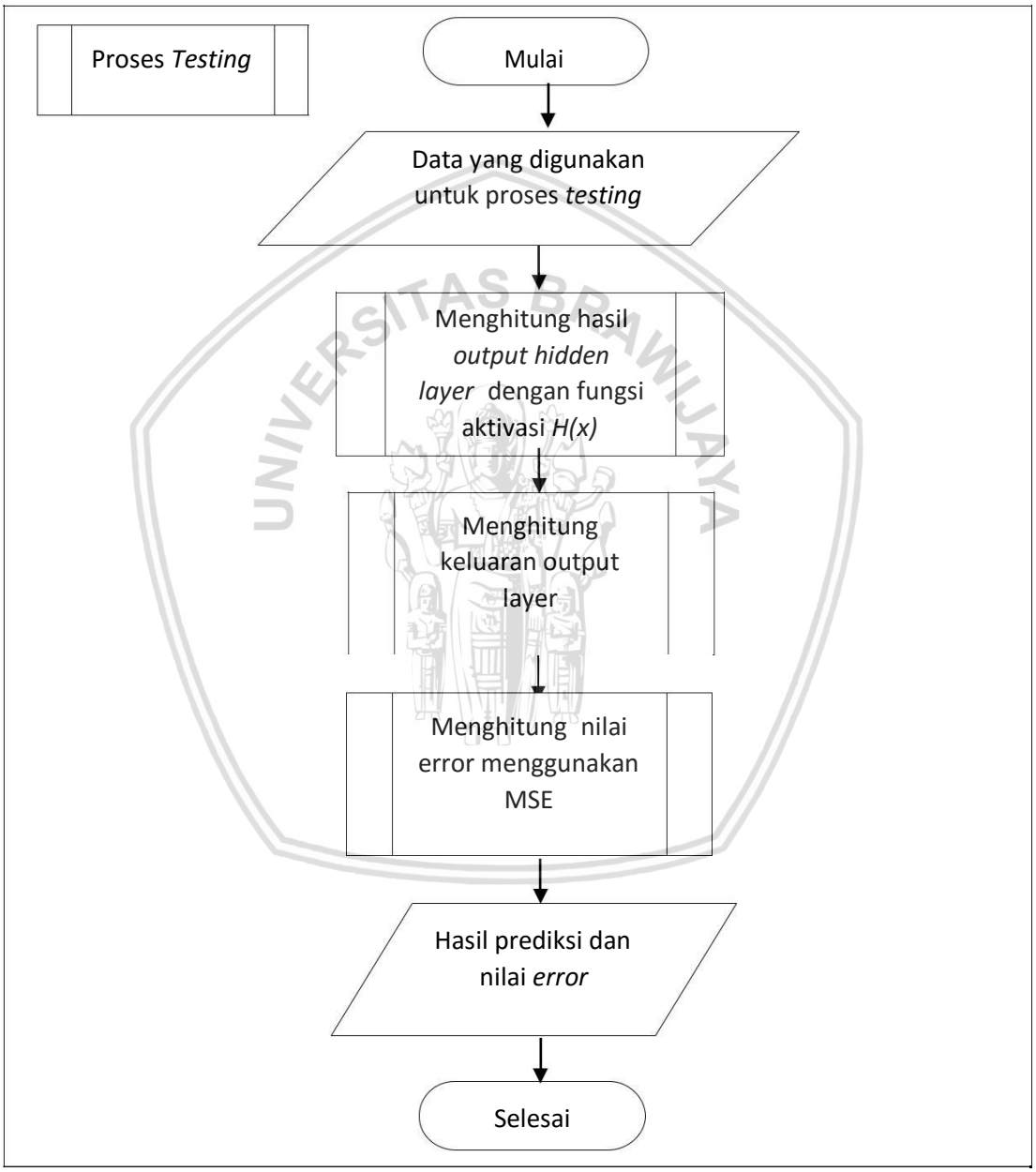
Gambar 4.6 *Flowchart* Menghitung *Output Weight*

Berdasarkan *flowchart* pada Gambar 4.6 telah dijelaskan langkah-langkah menghitung *output weight* yaitu menggunakan matriks dari hasil *output hidden layer* yang dihitung hingga mendapatkan matriks *Moonre-Penrose*(H+) dan

matriks target. Matriks target didapatkan dari normalisasi data aktual prediksi penjualan roti pada Harum *Bakery*.

4.2.4 Proses Testing

Proses *testing* ini kurang lebih hampir sama dengan proses *Training*. Proses ini menggunakan nilai yang didaptakan dari proses *training* yaitu nilai *input weight*, bias, dan hasil *output weight*. Pada *flowchart* pada Gambar 4.7 akan ditunjukkan langkah-langkah dari proses *testing*.

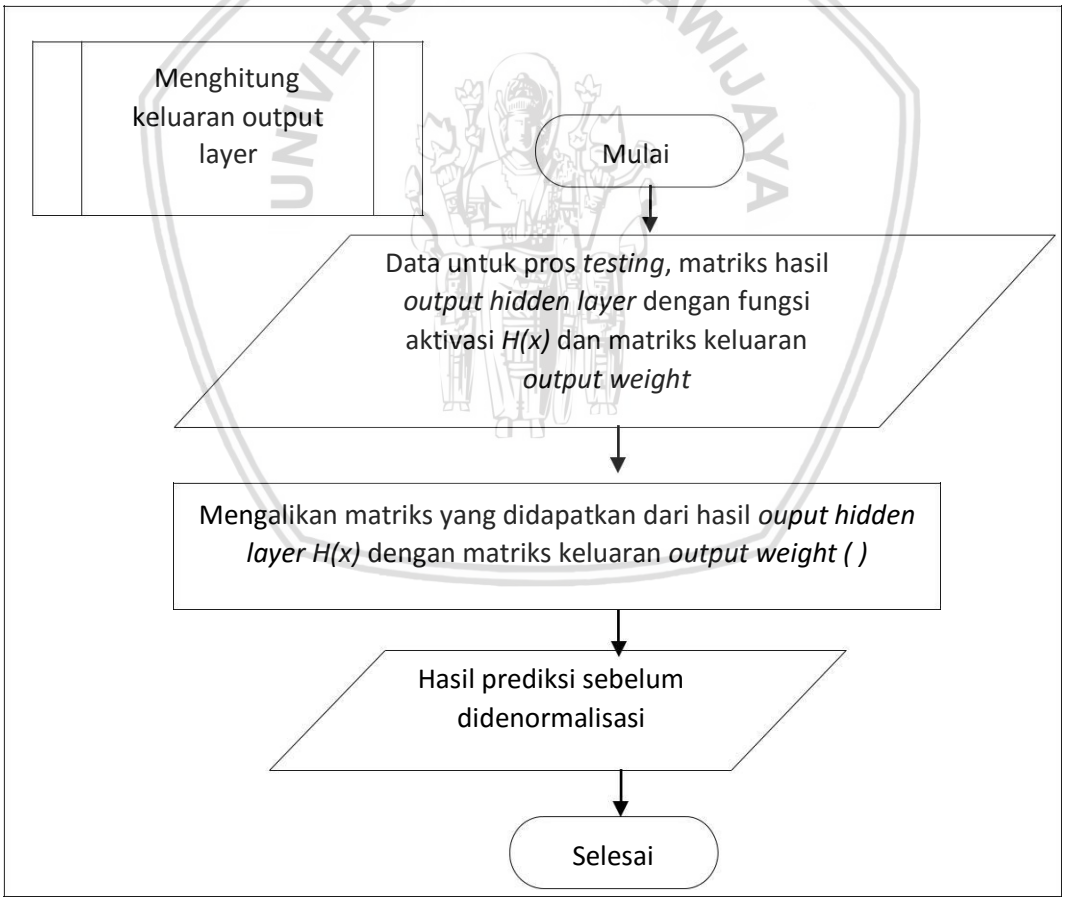


Gambar 4.7 *Flowchart* Proses *Testing*

Berdasarkan *Flowchart* proses *Training* pada Gambar 4.7, akan dijelaskan langkah-langkah untuk melakukan proses *testing* dimana langakah-langkahnya sebagai berikut :

1. Masukkan yang digunakan dalam proses ini adalah data yang digunakan sebagai data *testing*, fungsi aktivasi sigmoid, dan jumlah *neuron*.
2. *Output hidden layer* dihitung dengan fungsi aktivasi sigmoid yang telah di tunjukkan dan dijelaskan pada *Flowchart* pada Gambar 4.5. hasil dari perhitungan ini nanti akan befungsi untuk menghitung nilai dari *output layer* seperti pada *flowchart* pada Gambar 4.8.
3. Setelah mendapatkan nilai dari output layer, maka langkah berikutnya adalah menghitung nilai error yang mana pada penelitian ini menggunakan *Mean Square Error* (MSE) sebagai metode perhitungan error nya.
4. Dari proses *Testing* ini maka didapatkan Hasil prediksi yang belum dinormalisasi dan hasil dari nilai error hasil dari perhitungan MSE nya.

4.2.4.1 Proses Menghitung Keluaran Output Layer



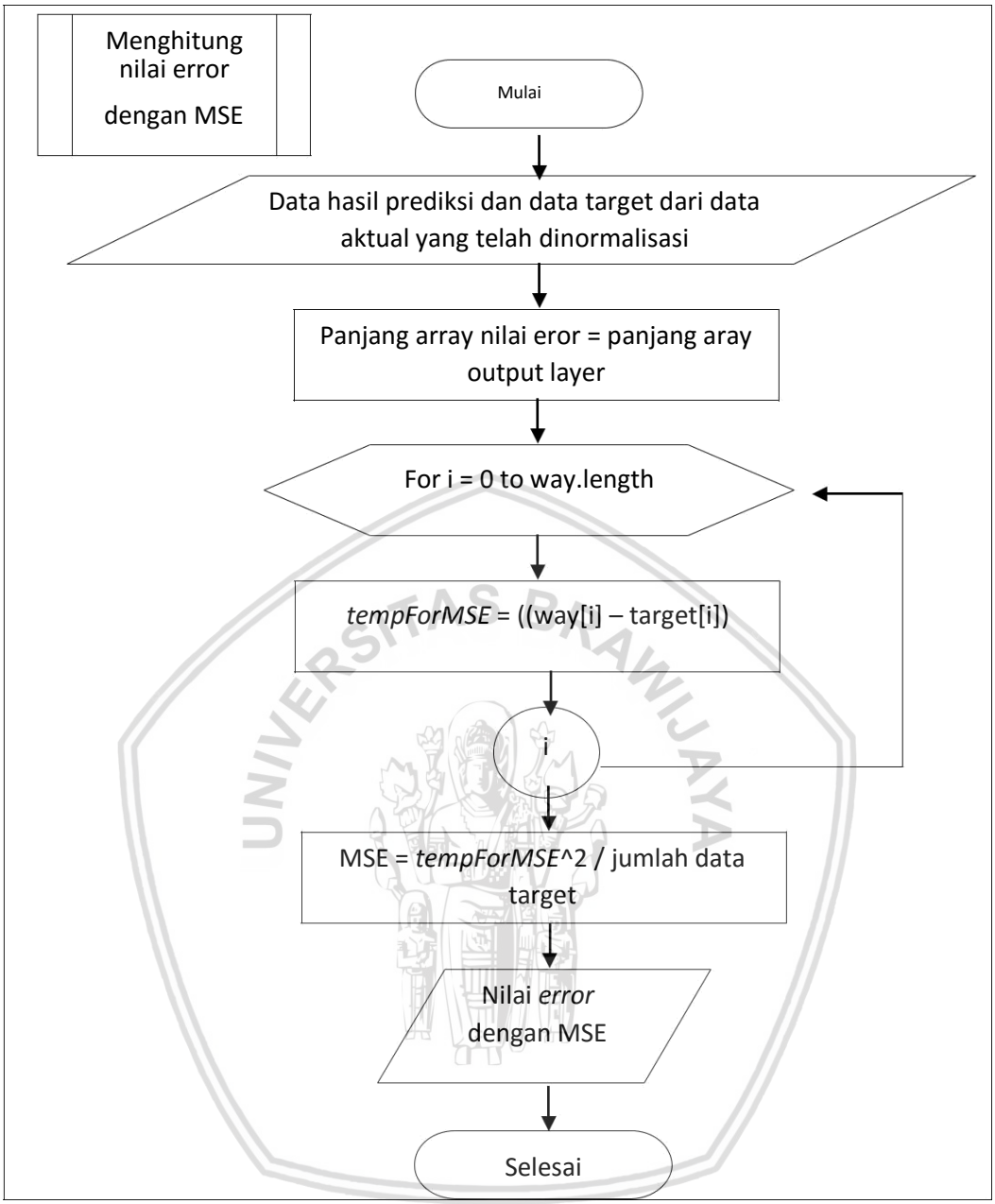
Gambar 4.8 *Fowchart* Proses Menghitung Keluaran *Output Layer*

Berdasarkan *Flowchart* proses *Training* pada Gambar 4.8, dijelaskan langkah-langkah untuk melakukan proses menghitung keluaran *output layer* dimana langakah-langkahnya sebagai berikut :

1. Masukkan yang diinputkan ke sistem yaitu hasil dari proses sesbelum sebelumnya yaitu matriks *output hidden layer* untuk data pada proses testing dan matriks *output weight*.
2. Kemudian pada proses berikutnya adalah matriks *output hidden layer* dengan fungsi aktivasi (*H(x*)) dikalikan dengan matriks output weight sesuai pada Persamaan 2.7.
3. Sistem memberi output hasil prediksi sebelum dinormalisasi yaitu berupa hasil dari proses *testing*.

4.2.4.2 Proses Hitung Nilai *Error* Menggunakan Metode *Mean Square Error* (MSE)

Penelitian ini menggunakan *Mean Square Error (*MSE) untuk mengevaluasi hasil prediksi menggunakan metode *Extreme Learning Machine*. Pada Gambar 4.9 ditunjukkan *Flowchart* dari proses hitung nilai *Mean Square Error* (MSE).



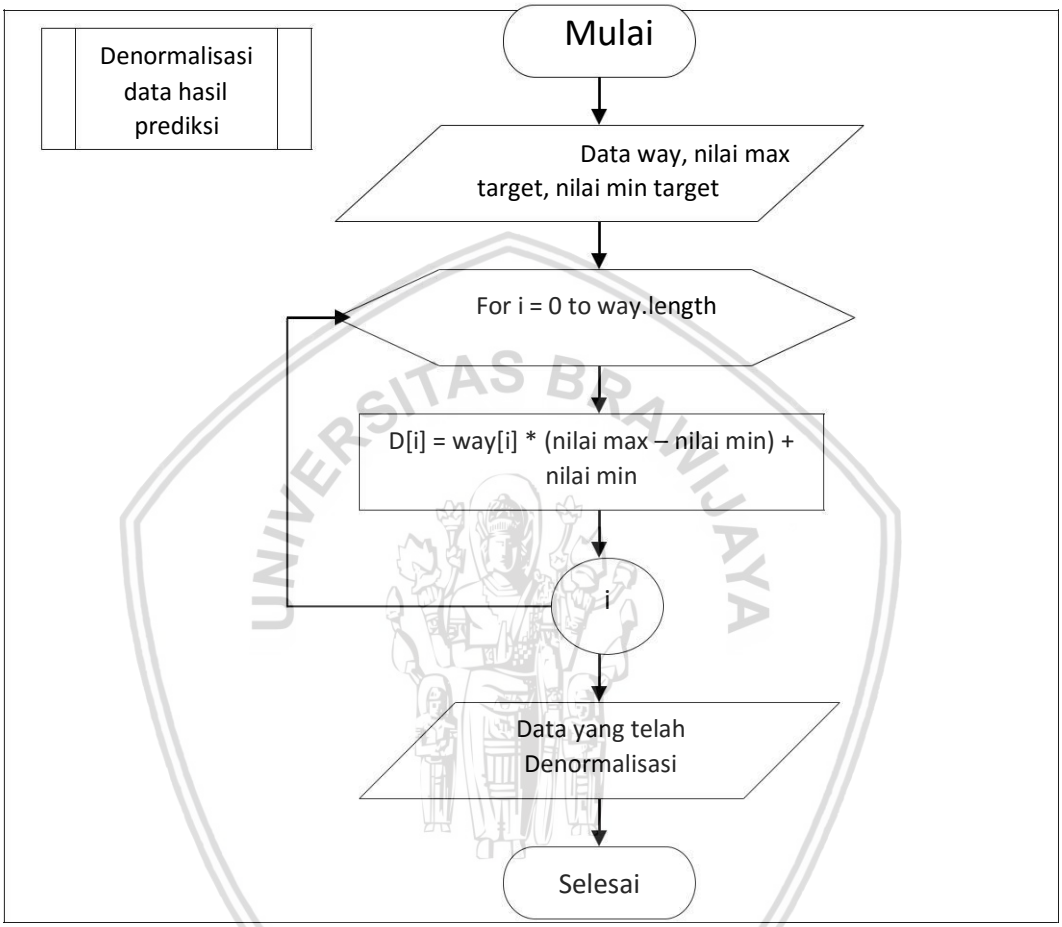
Gambar 4.9 *Flowchart* Proses Hitung Nilai *Error* Menggunakan MSE

Berdasarkan *Flowchart* proses hitung nilai *error* menggunakan MSE pada Gambar 4.9, dijelaskan langkah-langkah untuk melakukan proses hitung nilai *error* menggunakan MSE, langakah-langkahnya sebagai berikut :

1. Masukkan yang digunakan adalah hasil dari proes *testing* yang merupakan hasil prediksi yang belum terdenormalsasi, data target, dan jumlah dari kesluruhan data.
2. Dari data masukkan tersebut, maka data tersebut diolah dengan rumus MSE pada Persamaan 2.9 sehingga didapatkan nilai MSE
3. *Output* dari sistem berupa nilai *error* dengan MSE.

4.2.5 Proses Denormalisasi

Pada proses ini, berfungsi untuk mengubah kembali data yang sudah dinormalisasi menjadi data normal, dimana hasil prediksi dari hasil *testing* tersebut diubah menjadi data yang sebenarnya. Pada *flowchart* Gambar 4.10 ditunjukkan lankah-langkah untuk melakukan proses denormalisasi.



Gambar 4.10 *Flowchart* Proses Denormalisasi Data

Berdasarkan *flowchart* proses denormalisasi data pada Gambar 4.10, dijelaskan langkah-langkah untuk melakukan proses denormalisasi data, langakah-langkahnya sebagai berikut :

1. Masukkan yang dipakai dalam proses ini adalah nilai hasil prediksi sebelum didenormalisasi, nilai max, dan nilai min.
2. Selanjutnya data masukkan akan diproses menggunakan Persamaan 2.8.
3. Proses ini berfungsi untuk mengembalikan nilai yang ternormalisasi menjadi nilai yang sebenarnya.

4.3 Perhitungan Manual

Pada perhitungan manual ini, peneliti memakai 10 data yang terdiri dari 14 hari penjualan roti sebagai samplenya dan fungsi aktivasi sigmoid sebagai fungsi aktivasinya. Penelitian ini menggunakan data historis yaitu data penjualan roti 4 hari sebelum hari yang akan diprediksi. Data *training* sebesar 80% dan data *testing* sebesar 20% dari data akan digunakan dalam penelitian ini. Jadi dengan 10 data tersebut, maka data akan dibagi dua yaitu 8 data untuk data *training* dan 2 data untuk data *testing* nya.

4.3.1 Inisialisasi fitur yang digunakan

Tabel 4.2, Tabel 4.3, dan Tabel 4.4 merupakan sampel dari data dimana perhitungan manual ini menggunakan proses *learning* dimana akan menggunakan data 4 hari penjualan roti manis, roti tawar, dan roti cake sebelum hari yang akan diprediksi dan data target prediksi penjualan roti.

Tabel 4.2 *Data* Penjualan Roti Manis

manis	X1	X2	X3	X4	T
1	257	101	136	136	121
2	101	136	136	121	165
3	136	136	121	165	139
4	136	121	165	139	154
5	121	165	139	154	230
6	165	139	154	230	224
7	139	154	230	224	157
8	154	230	224	157	320
9	230	224	157	100	122
10	224	157	100	320	148

Tabel 4.3 *Data* Penjualan Roti Tawar

tawar	X1	X2	X3	X4	T
1	5	4	10	17	11
2	4	10	17	11	19
3	10	17	11	19	12
4	17	11	19	12	16

5	11	19	12	16	16
6	19	12	16	16	4
7	12	16	16	4	20
8	16	16	4	20	10
9	16	4	20	10	8
10	4	20	10	8	11

Tabel 4.4 *Data* Penjualan Roti Cake

cake	X1	X2	X3	X4	T
1	15	25	13	9	14
2	25	13	9	14	12
3	13	9	14	12	11
4	9	14	12	11	15
5	14	12	11	15	15
6	12	11	15	15	13
7	11	15	15	13	17
8	15	15	13	17	10
9	15	13	17	10	11
10	13	17	10	11	14

Keterangan:
X1 = data historis penjualan 4 hari sebelumnya
X2 = data historis penjualan 3 hari sebelumnya
X3 = data historis penjualan 2 hari sebelumnya
X4 = data historis penjualan 1 hari sebelumnya
T = data penjualan target prediksi

Seperti yang telah dipaparkan bahwa dari data tersebut akan dibagi 2 yakni 80% data *Training* dan 20% data *Testing.* Sehingga data ke-1 sampai ke-8 akan dijadikan sebagai data *Training* dan data ke 9 dan 10 akan dijadikan data *Testing* seperti pada Tabel 4.5, Tabel 4.6, Tabel 4.7, Tabel 4.8, Tabel 4.9 Tabel 4.10.

Tabel 4.5 Data *Training* roti Manis

manis	X1	X2	X3	X4	T
1	257	101	136	136	121

2	101	136	136	121	165
3	136	136	121	165	139
4	136	121	165	139	154
5	121	165	139	154	230
6	165	139	154	230	224
7	139	154	230	224	157
8	154	230	224	157	320

Tabel 4.6 Data *Testing* roti Manis

9	230	224	157	100	122
10	224	157	100	320	148

Tabel 4.7 Data *Training* roti Tawar

tawar	X1	X2	X3	X4	T
1	5	4	10	17	11
2	4	10	17	11	19
3	10	17	11	19	12
4	17	11	19	12	16
5	11	19	12	16	16
6	19	12	16	16	4
7	12	16	16	4	20
8	16	16	4	20	10

Tabel 4.8 Data *Testing* roti Tawar

9	16	4	20	10	8
10	4	20	10	8	11

Tabel 4.9 Data *Training* roti Cake

cake	X1	X2	X3	X4	T
1	15	25	13	9	14
2	25	13	9	14	12

3	13	9	14	12	11
4	9	14	12	11	15
5	14	12	11	15	15
6	12	11	15	15	13
7	11	15	15	13	17
8	15	15	13	17	10

Tabel 4.10 Data *Testing* roti Cake

9	15	13	17	10	11
10	13	17	10	11	14

Pada perhitungan manualisasi ini, peneliti menggunakan 2 *neuron* dan menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya dan menggunakan metode *Extreme Learning Machine* (ELM) sebagai metode pembelajarannya.

4.3.2 Proses normalisasi data

Min-Max Normalization merupakan metode yang peneliti gunakan untuk melakukan proses normalisasi data yang rumusnya seperti pada Persamaan 2.3. mulai dari data *Training* dan data *Testing* akan dilakukan proses normalisasi ini. Tabel 4.11 adalah nilai min dan maks pada data historis roti manis, tawar, dan roti cake. Ditahap ini peneliti melakukan pencarian nilai terbesar dan terkecil dari setiap fitur.

Tabel 4.11 Nilai Min Dan Maks pada data histioris Roti manis, tawar, dan cake

Manis	Max	Min
X1	257	101
X2	224	101
X3	224	100
X4	320	100
T	320	100
Tawar	Max	Min
X1	4	19
X2	4	20
X3	4	20

X4	4	20
T	4	20
Cake	Max	Min
X1	9	25
X2	9	25
X3	9	17
X4	9	17
T	9	17

Setelah mendapatkan nilai terkecil (min) dan nilai terbesar (max) pada setiap fitur didalam data historis penjualan roti manis, roti tawar, dan roti cake seperti pada Tabel 4.11, maka, data tersebut akan digunakan untuk melakukan normalisasi data yang disini kita menggunakan *Min-Max Normalization* sebagai metode untuk penormalisasian data sesuai dengan Persamaan 2.3 sehingga didapatkan hasil normalisasi pada Tabel 4.12, Tabel 4.13, dan Tabel 4.14.

Tabel 4.12 Hasil Normalisasi Data pada Roti Manis

manis	X1	X2	X3	X4	T
1	1	0	0.290322581	0.163636364	0.095454545
2	0	0.284552846	0.290322581	0.095454545	0.295454545
3	0.224358974	0.284552846	0.169354839	0.295454545	0.177272727
4	0.224358974	0.162601626	0.524193548	0.177272727	0.245454545
5	0.128205128	0.520325203	0.314516129	0.245454545	0.059090909
6	0.41025641	0.308943089	0.435483871	0.059090909	0.563636364
7	0.243589744	0.430894309	0.10483871	0.563636364	0.259090909
8	0.33974359	0.097560976	1	0.259090909	0
9	0.076923077	1	0.459677419	0	1
10	0.788461538	0.455284553	0	1	0.218181818

Tabel 4.13 Hasil Normalisasi Data pada Roti Tawar

tawar	X1	X2	X3	X4	T
1	0.066666667	0	0.375	0.8125	0.4375
2	0	0.375	0.8125	0.4375	0.9375
3	0.4	0.8125	0.4375	0.9375	0.5
4	0.866666667	0.4375	0.9375	0.5	0.75
5	0.466666667	0.9375	0.5	0.75	0.75
6	1	0.5	0.75	0.75	0
7	0.533333333	0.75	0.75	0	1
8	0.8	0.75	0	1	0.375
9	0.8	0	1	0.375	0.25
10	0	1	0.375	0.25	0.4375

Tabel 4.14 Hasil Normalisasi Data pada Roti Cake

cake	X1	X2	X3	X4	T
1	0.375	1	0.5	0	0.571428571
2	1	0.25	0	0.625	0.285714286
3	0.25	0	0.625	0.375	0.142857143
4	0	0.3125	0.375	0.25	0.714285714
5	0.3125	0.1875	0.25	0.75	0.714285714
6	0.1875	0.125	0.75	0.75	0.428571429
7	0.125	0.375	0.75	0.5	1
8	0.375	0.375	0.5	1	0
9	0.375	0.25	1	0.125	0.142857143
10	0.25	0.5	0.125	0.25	0.571428571

4.3.3 Inisialisasi *input weight* dan *bias*

Pada perhitungan manual ini, peneliti menggunakan 2 *neuron*. Inisialisasi *input weight* dan bias memiliki *range* dengan nilai 1 sampai dengan -1 yang nilai tersebut didapatkan secara acak atau *random*. Peneliti menggunakan matriks nilai *input weight* sebesar 2x4 dan *input weight* ini digunakan sebagai perhitungan pada ketiga jenis roti yang akan ditunjukkan pada Tabel 4.15.

Tabel 4.15 Matriks Nilai *Input Weight*

w	1	2	3	4
1	0.970588	-0.41618	-0.77892	-0.20598
2	-0.2936	0.274177	0.770641	0.717264

Jumlah *bias* dan *hidden neuron* adalah sama. Karena jumlah *hidden neuron* nya adalah 2, maka, jumah biasnya juga berjumlah dua yang jumlah bias ini juga diambil secara acak atau *random* seperti pada Tabel 4.16 matriks nilai bias.

Tabel 4.16 Matriks Nilai Bias

1	2
0.300383	-0.81216

4.3.4 Perhitungan Manual Proses *Training*

Pada manualisasi ini, peneliti menggunakan 2 neuron pada *hidden layer* dan fungsi aktivasi sigmoid sebagai fungsi aktivasinya. Pada proses training ini, peneliti menggunakan 80% dari data, yaitu 8 buah data, dimana 20% dari data lainnya adalah untuk proses *testing*. Berikut adalah langkah-langkah dari proses perhitungan manual proses *Training* :

4.3.4.1 Menghitung keluaran *hidden layer* dengan fungsi aktivasi

Langkah awal dalam proses *training* ini adalah menghitung keluaran *hidden layer* ( ) dengan rumus seperti Persamaan 2.4. Setelah seluruh proses perhitungan keluaran *hidden layer* selesai, maka nilai dari perhitungan itu diproses menggunakan fungsi aktivasi sigmoid (H(x)) sesuai dengan Persamaan 2.2. Berikut adalah contoh dari perhitungannya :

(Σ)

Roti Manis :

Tabel 4.17, Tabel 4.19, dan Tabel 4.20 adalah hasil dari proses perhitungan keluaran *Hidden layer* sebelum dihitung dengan fungsi aktivasi sigmoid.

Tabel 4.17 Matriks Keluaran *Hidden Layer* Roti Manis

	1	2
1	1.011127117	-0.764655043
2	-0.063841995	-0.441941562
3	0.206946327	-0.457583258
4	0.005652108	-0.302333783
5	-0.087273215	-0.288705089
6	0.218618372	-0.469920685
7	0.159719903	-0.280467559
8	-0.242756421	0.071317839

Setelah nya sudah diketahui, maka matriks keluaran *hidden layer* tersebut diproses menggunakan fungsi aktivasi sigmoid, yang mana perhitungannya dijelaskan berikut ini :

Tabel 4.18**,** Tabel 4.20, dan Tabel 4.22 adalah hasil dari perhitungan keluaran *hidden layer* dengan fungsi aktivasi sigmoid.

Tabel 4.18 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Manis

H(x)	1	2
1	0.73324067	0.31763646
2	0.48404492	0.391278431
3	0.551552726	0.387559299
4	0.501413023	0.42498707
5	0.478195534	0.428320912
6	0.554437948	0.384635017

7	0.539845306	0.430339151
8	0.439607186	0.517821907

Roti Tawar :

Tabel 4.19 Matriks Keluaran *Hidden Layer* Roti Tawar

	1	2
1	-0.094364883	0.230605188
2	-0.57867325	0.31009925
3	-0.2076765	0.150350375
4	0.0736795	0.239699438
5	-0.2090985	0.16560725
6	0.26354425	-0.1753465
7	-0.110648	-0.10946325
8	0.5587384	0.007255

Tabel 4.20 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Manis

H(x)	1	2
1	0.47642627	0.557397163
2	0.359237936	0.576909487
3	0.448266678	0.537516947
4	0.518411547	0.559639579
5	0.447915009	0.541307448
6	0.565507346	0.456275349
7	0.472366188	0.47266148
8	0.63616058	0.501813742

Roti Cake :

Tabel 4.21 Matriks Keluaran *Hidden Layer* Roti Cake

	1	2
1	-0.1412865	-0.764655043

2	1.0381885	-0.441941562
3	-0.0210375	-0.457583258
4	-0.17326325	-0.302333783
5	0.176443	-0.288705089
6	-0.30832925	-0.469920685
7	-0.421541	-0.280467559
8	-0.087154	0.071317839

Tabel 4.22 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Cake

H(x)	1	2
1	0.464737015	0.31763646
2	0.738500325	0.391278431
3	0.494740819	0.387559299
4	0.456792225	0.42498707
5	0.543996667	0.428320912
6	0.423522602	0.384635017
7	0.396148061	0.430339151
8	0.478225281	0.517821907

Setelah didapatkan nilai matriks keluaran *hidden* layer dengan fungsi aktivasi sigmoid maka matriks tersebut bisa digunakan kedalam perhitungan *output weight.* Tapi sebelum ke tahap output weight, masih ada tahap yang lain yang akan dibahas manualisasi sederhananya seperti transpose matriks keluaran *Hidden Layer* dengan fungusi aktivasi sigmoid, perkalian matrik transpose dengan matriks keluaran *hidden layer* dengan fungsi aktivasi, dan masih ada beberapa proses yang akan dijelaskan di bawah ini :

4.3.4.2 Mencari nilai matriks Moore-Penrose Generalized Invers

Langkah awal yang dilakukan adalah dengan mengkalikan matriks keluaran *hidden layer* yang telah diproses dengan fungsi aktivasi sigmoid yang sudah ditranspose kemudian dikalikan dengan mengkalikan matriks keluaran *hidden layer* yang telah diproses dengan fungsi aktivasi sigmoid yang belum ditranspose. Kemudian, hasil dari perkalian itu diinverskan, kemudian hasil dari matriks yang sudah diinvers itu kemudian dikalikan lagi dengan matriks keluaran *hidden layer* yang telah diproses dengan fungsi aktivasi sigmoid yang kemudian hasil dari

perkalian ini menjadi matriks *Moore-Penrose Generalized Invers* ( ). Tabel 4.23 merupakan hasil daru keluaran *hidden layer* dengan fungsi aktivasi yang telah ditranspose.

Tabel 4.23 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi pada Roti Manis yang Sudah Ditranspose

H^T	1	2	3	4	5	6	7	8
1	0.7332 4067	0.484044 92	0.551552 726	0.5014 13023	0.4781 95534	0.5544 37948	0.5398 45306	0.4396 07186
2	0.3176 3646	0.391278 431	0.387559 299	0.4249 8707	0.4283 20912	0.3846 35017	0.4303 39151	0.5178 21907

Setelah matriks keluaran *hidden layer* dengan fungsi aktivasi telah ditranspose seperti pada Tabel 4.23, maka langkahselanjutnta adalah mengkalikan keluaran *hidden layer* dengan fungsi aktivasi sigmoid ( ) dengan matriks keluaran *hidden layer* dengan fungsi aktivasi sigmoid (H(x)) yang hasilnya akan ditunjukkan pada Tabel 4.24, Tabel 4.25, dan Tabel 4.26.

Tabel 4.24 Hasil Perkalian Matriks Transpose Dengan Matriks Keluaran Hidden Layer Dengan Fungsi Aktivasi Pada Roti Manis

	1	2
1	2.348326634	1.727185946
2	1.727185946	1.369542163

Tabel 4.25 Hasil Perkalian Matriks Transpose Dengan Matriks Keluaran Hidden Layer Dengan Fungsi Aktivasi Pada Roti Tawar

	1	2
1	1.973983944	2.046871195
2	2.046871195	2.222063934

Tabel 4.26 Hasil Perkalian Matriks Transpose Dengan Matriks Keluaran Hidden Layer Dengan Fungsi Aktivasi Pada Roti Cake

	1	2
1	2.075727312	1.636469182
2	1.636469182	1.369542163

Setelah hasil dari perkalian antara matriks keluaran *hidden layer* dengan fungsi aktivasi yang sudah ditranspose dengan matriks keluaran *hidden layer* dengan fungsi aktivasi yang belum ditranspose sudah ada, maka matriks dari perkalian tersebut diinverskan yang hasilnya seperti pada Tabel 4.27, Tabel 4.28, dan Tabel 4.29

Tabel 4.27 Invers Matriks Roti Manis

( )	1	2
1	5.878846216	-7.414054738
2	-7.414054738	10.08034036

Tabel 4.28 Invers Matriks Roti Tawar

( )	1	2
1	11.30034404	-10.40939838
2	-10.40939838	10.03872902

Tabel 4.29 Invers Matriks Roti Cake

( )	1	2
1	8.312109689	-9.932159601
2	-9.932159601	12.59813211

kemudian hasil matriks invers (( )) tersebut dikalikan dengan matriks keluaran *hidden layer* dengan fungsi aktivasi sigmoid yang telah di transpose ( ). Keluaran dari perkalian tersebut disebut dengan *Moore-Penrose Generalized Invers* ( ) sesuai dengan Persamaan 2.5 dan hasil dari perhitungannya dapat dilihat pada Tabel 4.30, Tabel 4.31, dan Tabel 4.32

( )

( )

( )

Tabel 4.30 Matriks *Moore-Penrose Generalized Invers* Roti Manis

	1	2	3	4	5	6	7	8
1	1.955635035	-0.055334062	0.369107805	-0.203147344	-0.364356681	0.407750366	-0.016890492	-1.254776917
2	-2.2344	0.355484233	-0.1825	0.566510713	0.772252713	-0.2333	0.335522472	1.960549324

	02832		12464			81414		

Tabel 4.31 Matriks *Moore-Penrose Generalized Invers* Roti Tawar

	1	2	3	4	5	6	7	8
1	-0.4183 88367	-1.9457 68407	-0.5296 60358	0.0327 17503	-0.5730 91168	1.6408 75695	0.4177 78789	1.9652 54268
2	0.6362 48237	2.0519 87219	0.7298 00548	0.2217 17772	0.7715 13018	-1.3061 66669	-0.1721 27311	-1.5844 76736

Tabel 4.32 Matriks *Moore-Penrose Generalized Invers* Roti Cake

	1	2	3	4	5	6	7	8
1	0.7081 29028	2.2522 55875	0.2630 39146	-0.4241 32323	0.2676 08303	-0.2998 90047	-0.9813 70997	-1.1680 28828
2	-0.6142 16118	-2.4055 25721	-0.0313 21531	0.8171 09963	-0.0070 18279	0.6391 88672	1.4868 63712	1.7737 78969

4.3.4.3 Mencari nilai *Output Weight*

Langkah yang terakhir pada proses *Training* adalah menghitung *output weight.* Langkah-langkah dalam menghitung *Output weight* adalah sesuai dengan Persamaan 2.6, dimana perhitungan *output* weight ini mengkalikan matriks *Moore-Penrose Generalized Invers* ( ) dengan target dari data *Training* pada Tabel 4.13, Tabel 4.14, dan Tabel 4.15 yang mana hasil dari perhitugan dapat dilihat di Tabel 4.33, Tabel 4.34, dan Tabel 4.35.

( ) ( )

Tabel 4.33 Nilai *Output Weight* Roti Manis

0.389811455
-0.000534641

Tabel 4.34 Nilai *Output Weight* Roti Tawar

-1.522564081
2.545613902

Tabel 4.35 Nilai *Output Weight* Roti Cake

-0.135974316
1.296690425

4.3.5 Perhitungan Manual Proses *Testing*

Perhitungan manual proses *testing* ini berfungsi untuk mendapatkan hasil prediksi dengan menggunakan metode *Extreme Learning Machine* yang dihasilkan dari beberapa proses pada proses *training* seperti perkalian matriks *hidden layer* dengan fungsi aktivasi, matriks *Moore-Penrose Generalized Invers* dan yang kemudian pada akhirnya akan menghasilkan *output weight* yang nantinya akan digunakan pada proses *testing*. Pada perhitungan manual proses *testing* ini memggunakan 20% dari data yang telah ditentukan diatas, oleh karena itu proses *testing* ini menggunakan data sejumlah 2 buah data. Proses testing ini mirip dengan proses *Training* seperti menggunakan fungsi aktivasi sigmoid, *input weight* dan bias yang telah didapatkan secara *random*, dan jumlah *hidden neuron* pada *hidden layer* sebanyak 2 hanya beda data saja yang diproses dan pada proses *testing* ini ada proses untuk mencari *output layer*. Pada Tabel 4.36, Tabel 4.37, dan Tabel 4.38, akan ditunjukkan hasil dari normalisasi data *testing* sebagai berikut :

Tabel 4.36 Normalisasi Data *Testing* Roti Manis

Data ke-	X1	X2	X3	X4	T
1	0.076923077	1	0.45967741	0	1
2	0.788461538	0.455284553	0	1	0.218181818

Tabel 4.37 Normalisasi Data *Testing* Roti Tawar

Data ke-	X1	X2	X3	X4	T
1	0.8	0	1	0.375	0.25
2	0	1	0.375	0.25	0.4375

Tabel 4.38 Normalisasi Data *Testing* Roti Cake

Data ke-	X1	X2	X3	X4	T
1	0.375	0.25	1	0.125	0.142857143

2	0.25	0.5	0.125	0.25	0.571428571

4.3.5.1 Menghitung keluaran *hidden layer* dengan fungsi aktivasi

Setelah mendapatkan data yang telah ternormalisasi, maka langkah perhitungan yang dilakukan adalah menghitung keluaran *hidden layer* dengan fungsi aktivasi, caranya sama seperti pada saat melakukan proses *training.*

Roti Manis :

Tabel 4.39, Tabel 4.40, dan Tabel 4.41 merupakan hasil dari perhitungan keluaran *hidden layer* yang belom dihitung dengan fungsi aktivasi :

Tabel 4.39 Matriks Keluaran *Hidden Layer* Roti Manis

	1	2
9	-0.39918832	-0.206321349
10	0.670193982	-0.201559755

Tabel 4.40 Matriks Keluaran *Hidden Layer* Roti Tawar

	1	2
9	0.2206909	-0.007425
10	-0.459387	-0.069676625

Tabel 4.41 Matriks Keluaran *Hidden Layer* Roti Cake

	1	2
9	-0.244359	0.00658325
10	0.18608	-0.472825375

Setelah nya sudah diketahui, maka matriks keluaran *hidden layer* tersebut diproses menggunakan fungsi aktivasi sesuai dengan Persamaan 2.1, sama dengan tahapan pada proses *testing.*

Tabel 4.42, Tabel 4.43 dan Tabel 4.44 adalah hasil dari perhitungan keluaran *hidden layer* dengan fungsi aktivasi sigmoid.

Tabel 4.42 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Manis

H(x)	1	2
1	0.40150737	0.448601862
2	0.661546594	0.449779968

Tabel 4.43 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Tawar

H(x)	1	2
1	0.55494988	0.498143759
2	0.387131255	0.482587888

Tabel 4.44 Matriks Keluaran *Hidden Layer* Dengan Fungsi Aktivasi Roti Cake

H(x)	1	2
1	0.439212425	0.501645807
2	0.546386231	0.383947734

4.3.5.2 Menghitung Nilai *Output Layer*

Langkah terakhir dari proses *Testing* adalah menghitung nilai *output layer* nya dengan rumus pada Persamaan 2.7.

( ) ( )

Tabel 4.45, Tabel 4.46 dan Tabel 4.47 adalah hasil dari perhitungan dari *output layer*

Tabel 4.45 Keluaran *Output Layer* Roti Manis

0.156272331
0.257637969

Tabel 4.46 Keluaran *Output Layer* Roti Tawar

0.423134922
0.639050292

Tabel 4.47 Keluaran *Output Layer* Roti Cake

0.590757705
0.423566856

4.3.6 Perhitungan Nilai MSE

Pada proses perhitungan *mean square error*(MSE) ini berfungsi untuk menghitung seberapa besar error prediksi menggunakan metode *Extreme Learning Machine* yang mana pada proses ini menggunakan Persamaan 2.9 sebagai rumus perhitungannya. Berikut adalah contoh perhitungan dari MSE dan pada Tabel 4.47 akan ditunjukkan hasil dari nilai MSE nya.

Tabel 4.48 Keluaran *Output Layer* Roti Tawar

∑ ( )

Jenis Roti	MSE
Roti Manis	0.35671658
Roti Tawar	0.03529911
Roti Cake	0.111239

( ) ( )

4.3.7 Perhitungan Denormalisasi Data

Proses Denormalisasi data adalah sebuah perhitungan yang berfungsi untuk mengembalikan nilai data sesuai dengan nilai aslinya. Perhitungan ini menggunakan Persamaan 2.8 sebagai rumus perhitungan denormalisasi data. disini adalah *output layer* dan nilai dari min dan max dari rumus tersebut diambil dari nilai Min max fitur T.

(( ))

Tabel 4.49 adalah hasil dari *output layer* yang telah didenormalisasi yang kemudian ini adalah hasil dari prediksi penjualan roti menggunakan metode *Extreme Learning Machine*.

Tabel 4.49 Hasil Perhitungan Denormalisasi Data

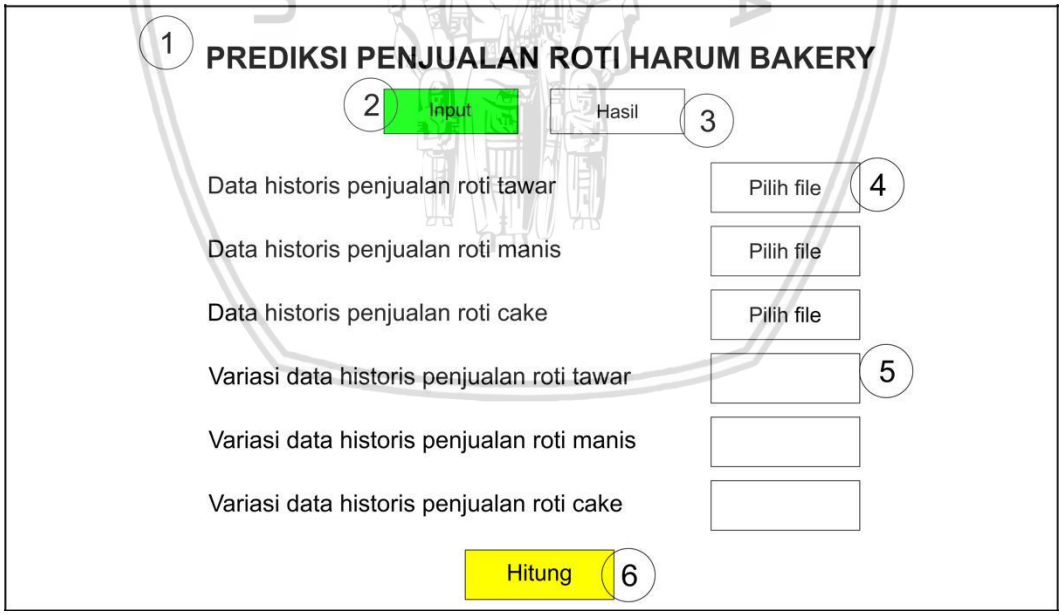
	Roti Manis	Roti Tawar	Roti Cake
Denormalisasi	134.3799129	10.77015875	14.13530394
	156.6803533	13.58575439	15.77706969

4.4 Perancangan Antarmuka

Pada bagian perancagan antarmuka ini berifungsi sebagai gambaran sistem prediksi penjualan roti pada Harum *Bakery* yang nantinya akan diimplementasikan pada bab implementasi. Terdapat beberapa tampilan dalam perancangan antarmuka ini, yaitu perancangan halaman *import* data untuk *user,* perancangan halaman normalisasi, perancangan halaman *training,* perancangan halaman *testing*, perancangan halaman denormalisasi, perancangan halaman *evaluasi* dan perancangan halaman *weight* & bias.

4.4.1 Perancangan Halaman *Import* Data

Halaman ini merupakan halaman *import* data dimana user memasukkan beberapa data seperti data historis penjualan untuk ketiga jenis roti, variasi data historis penjualan untuk ketiga jenis roti, dan memasukkan jumlah *hidden neuron.* Antarmuka halaman ini dirancang di Gambar 4.11.



Gambar 4.11 Rancangan Halaman *Import Data*

Keterangan dari Gambar 4.11 merupakan rancangan halaman import data adalah:

1. *Header* aplikasi.
2. *Tab* menu halaman yang *active.*

3. Tombol untuk *choose file* historis penjualan roti dengan format.xls.
4. *Text box* untuk menginputkan jumlah variasi data historis penjualan roti.
5. *Text box* untuk menginputkan jumlah *neuron* pada *hidden neuron.*
6. *Tab* untuk melakukan proses perhitungan dan menampilakan data historis.

4.4.2 Perancangan Halaman Normalisasi

Halaman ini merupakan halaman untuk menunjukkan hasil dari proses normalisasi dalam prediksi penjualan roti menggunakan ELM. Pada peneitian ini, prediksian ini menggunakan *min-max normalizaton*. Perancangan halaman normalisasi ini ditunjukkan pada Gambar 4.12.

Gambar 4.12 Rancangan Halaman Normalisasi

Keterangan :

1. *Header* aplikasi.
2. *Tab* menu yang *active* yaitu *tab* normalisasi.
3. Data *view* memperlihatkan proses dan hasil normalisasi dalam prediksi penjualan roti.

4.4.3 Perancangan Halaman *Training*

Halaman ini merupakan halaman untuk menunjukkan hasil dari proses *training* dalam prediksi penjualan roti menggunakan ELM. Perancangan halaman *training* ini ditunjukkan pada Gambar 4.13.

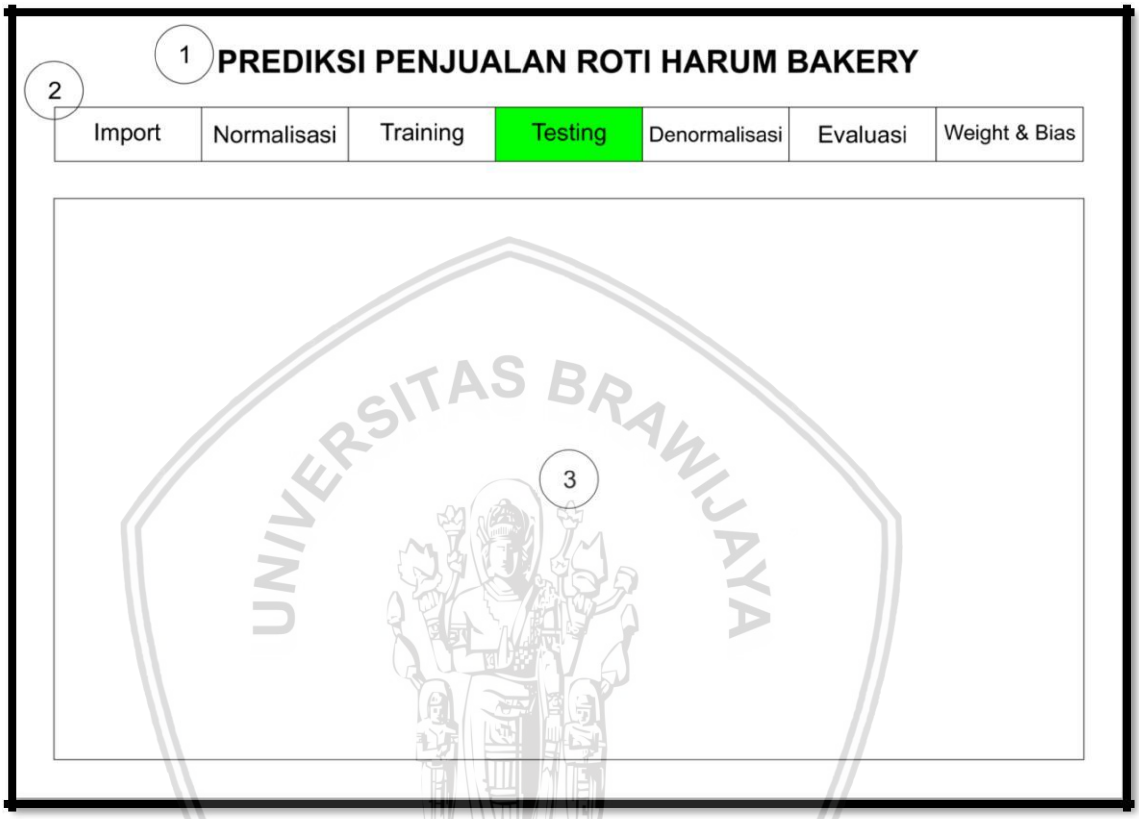
Gambar 4.13 Halaman *Training*

Keterangan :

1. *Header* aplikasi
2. *Tab* menu yang *active* yaitu *tab training.*
3. Data *view* memperlihatkan proses dan hasil *training* dalam prediksi penjualan roti.

4.4.4 Perancangan Halaman *Testing*

Halaman ini merupakan halaman untuk menunjukkan hasil dari proses *testing* dalam prediksi penjualan roti menggunakan ELM. Perancangan halaman *testing* ini ditunjukkan pada Gambar 4.14.



Gambar 4.14 Halaman *Testing*

Keterangan :

1. *Header* aplikasi.
2. *Tab* menu yang *active* yaitu *tab testing.*
3. Data *view* memperlihatkan proses dan hasil *testing* dalam prediksi penjualan roti.

4.4.5 Perancangan Halaman Denormalisasi

Halaman ini merupakan halaman untuk menunjukkan hasil dari proses denormalisasi dalam prediksi penjualan roti menggunakan ELM. Perancangan halaman denormalisasi ini ditunjukkan pada Gambar 4.15.

Gambar 4.15 Halaman Denormalisasi

Keterangan :

1. *Header* aplikasi.
2. *Tab* menu yang *active* yaitu *tab* denormalisasi.
3. *Data view* memperlihatkan proses dan hasil denormalisasi dalam prediksi penjualan roti.

4.4.6 Perancangan Halaman Evaluasi

Halaman ini merupakan halaman untuk menunjukkan hasil dari proses evaluasi prediksi, dimana dalam prediksi penjualan roti menggunakan ELM ini menggunakan metode MSE sebagai metode untuk mengevaluasi tingkat erornya. Perancangan halaman evaluasi ini ditunjukkan pada Gambar 4.16.

Gambar 4.16 Perancangan Halaman Evaluasi

Keterangan :

1. *Header* aplikasi.
2. *Tab* menu yang *active* yaitu *tab* evaluasi.
3. *Data view* memperlihatkan proses dan hasil dari evaluasi prediksi menggunakan MSE.

4.4.7 Perancangan Halaman *Input Weight* dan *Bias*

Halaman ini merupakan halaman untuk menunjukkan data *input weight* dan *bias* dalam prediksi penjualan roti menggunakan ELM. Perancangan halaman *input weight* dan *bias* ini ditunjukkan pada Gambar 4.17.

Gambar 4.17 Perancangan Halaman *Weight* dan Bias

Keterangan :

1. *Header* aplikasi.
2. *Tab* menu yang *active* yaitu *tab weight & bias.*
3. Data view memperlihatkan nilai *input weight & bias.*

4.5 Pengujian Algoritme

Pada bagian pengujian algoritma ini ditujukan untuk menguji metode *Extreme Learning* machine(ELM) dengan beberapa pengujian yang tujuannya adalah untuk mengetahui tingkat erornya menggunakan metode *Mean Square Error*(MSE). Adapun pengujian yang peneliti ujikan adalah sebagai berikut :

1. Pengujian jumlah *neuron* pada *hidden layer.*
2. Pengujian jumlah fitur data historis penjualan.
3. Pengujian jumlah data data historis penjualan.

4.5.1 Pengujian Jumlah Neuron pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* bertujuan untuk mengetahui berapa neuron kah yang palinyg baik digunakan jika berdasarkan nilai rata-rata *Mean Square Error* (MSE) (Huang, Zhu, & Siew, 2006). Pada pengujian ini, peneliti

menggunakan 1 sampai 7 jumlah *neuron* pada *hidden layer*. Data yang digunakan adalah data historis penjualan roti manis, roti tawar, dan roti cake selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang masing-masing memiliki total 91 data per jenis rotinya. Presentase pembagian data *Training* dan *Testing* disini dibagi sebesar 80% data *Training* dan 20% data *Testing* dengan jumlah fitur datanya sejumlah 4 fitur dan menggunakan fungsi aktivasi sigmoid. Setiapkali pengujian dengan *neuron n* maka akan dilakukan 10 kali percobaan, kemudian dihitung rata rata dari nilai MSE dari *neuron* n, dan dibuat grafik rata-rata nilai MSE nya. Tabel 4.50 merupakan tabel rancangan pengujian jumlah *neuron* pada *hidden layer*.

Tabel 4.50 Rancangan Pengujian Jumlah Neuron pada *Hidden Layer*

Percobaan ke-*i*	Nilai MSE pada Jumlah Hidden Neuron						
	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
Rata-Rata nilai MSE							

4.5.2 Pengujian Jumlah Fitur Data Historis Penjualan

Pengujian jumlah fitur data historis penjualan ini diuji untuk mendapatkan jumlah fitur historis penjualan yang terbaik yang diketahui setelah proses perhitungan nilai MSE telah dilakukan. Pada pengujian ini, peneliti menggunakan 4 fitur hingga 9 fitur sebagai pengujiannya. Maksud dari 4 fitur adalah perhitungan prdiksi penjualan roti berdasarkan 4 hari sebelumya, 4 fitur adalah perhitungan prediksi penjualan roti berdasarkan 4 hari sebelumnya, dan hingga sampai 9 fitur yang maksudnya adalah perhitungan prediksi penjualan roti berdasarkan 9 hari sebelumnya. Data yang digunakan adalah data historis penjualan roti manis, roti tawar, dan roti cake selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang masing-masing memiliki total 91 data per

jenis rotinya. Presentase pembagian data *Training* dan *Testing* disini dibagi sebesar 80% data *Training* dan 20% data *Testing* dan menggunakan fungsi aktivasi sigmoid. Setiapkali pengujian dengan fitur n maka akan dilakukan 10 kali percobaan, kemudian dihitung rata rata dari nilai MSE dari fitur n, dan dibuat grafik rata-rata nilai MSE nya. Tabel 4.51 merupakan tabel jumlah fitur data historis penjualan.

Tabel 4.51 Rancangan Pengujian Jumlah Fitur Data Historis Penjualan

Percobaan ke-*i*	Nilai MSE pada Jumlah Fitur Data Historis Penjualan				
	4	5	6	7	8
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Rata-rata Nilai MSE					

4.5.3 Pengujian Jumlah Data Historis Penjualan

Pengujian jumlah data historis penjualan bertujuan untuk menguji apakah semakin banyak data yang didapat berbanding lurus dengan nilai *Mean Square Error* (MSE) yang dihasilkan. Pada penujian ini akan menggunakan data historis penjualan roti manis, roti tawar, dan roti cake selama 1 bulan hingga data historis penjualan selama 5 bulan yang perbulannya memilki data sejumlah 30 hingga 31 data. Presentase pembagian data *Training* dan *Testing* disini dibagi sebesar 80% data *Training* dan 20% data *Testing* dengan jumlah fitur datanya sejumlah 4 fitur dan menggunakan fungsi aktivasi sigmoid. Setiapkali pengujian dari bulan 1 hingga 5 akan dilakukan 10 kali percobaan, kemudian dihitung rata rata dari nilai MSE dari bulan 1 hingga 5, dan dibuat grafik rata-rata nilai MSE nya. Tabel 5.2 merupakan tabel rancangan pengujian jumlah data.

Tabel 4.52 Rancangan Pengujian Jumlah Data

Percobaan ke-*i*	Nilai MSE pada Jumlah Data *Training*				
	Bulan ke-1	Bulan ke-2	Bulan ke-3	Bulan ke-4	Bulan ke-5
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Rata-rata nilai MSE					

BAB 5 IMPLEMENTASI

Setelah melakukan perancangan dan pengujian pada bab 4, pada bab Implementasi pada bab 5 ini akan membahas tentang implementasi sistem dan antarmuka yang telah dirancang pada bab perancangan dan pembahasan prediksi penjualan roti menggunakan metode ELM.

5.1 Implementasi Sistem

pada subab implementasi sistem ini, berisi tentang pembahsaan implementasi sistem berupa *coding* yang mencantumkan proses dari Implementasi metode *Extreme Learning Macine* (ELM) untuk memprediksikan penjualan roti pada Harum *Bakery.*

5.1.1 Implementasi Proses Normalisasi Data

pada implementasi proses normalisasi data ini menggunakan seluruh data, dengan fitur yang telah ditentukan, maka sistem akan mengambil nilai min dan max dari setiap fitur pada data awal. Kemudian data awal akan dinormalisasi yang kemudian data yang ternormalisasi tersebut akan digunakan pada proses *training* dan *testing.* Gambar 5.1 merupakan implementasi proses normalisasi data.

```
1   //Inisialisasi dataAwal dan dataNormalisasi
2   this.dataAwal = new double[this.dataValue.size() -
3   (this.variation)][this.variation + 1];
4   this.dataNormalisasi = new double[this.dataValue.size() -
5   (this.variation)][this.variation + 1];
6
7
8   // Generate dataAwal
9   for (int i = 0; i < dataAwal.length; i++) {
10     for (int k = 0; k < dataAwal[0].length; k++) {
11        dataAwal[i][k] = this.dataValue.get(i + k);
12     }
13  }
14
15
16  // Mencari Nilai Maksimal
17  seekingMax(dataAwal);
18
19  //fungsi cari nilai max
20  public void seekingMax(double[][] value) {
21
22     maxValue = new double[value[0].length];
```

```
23
24      for (int i = 0; i < value[0].length; i++) {
25          double temp = 0.0;
26          for (int k = 0; k < value.length; k++) {
27              temp = cekMax(temp, value[k][i]);
28          }
29          this.maxValue[i] = temp;
30      }
31  }
32
33  //fungsi cek nilai max untuk perbandingan pada perulangan
34      public double cekMax(double value1, double value2) {
35
36          if (value1 > value2) {
37              return value1;
38          } else {
39              return value2;
40          }
41      }
42
43
44  // Mencari Nilai Minimal
45  seekingMin(dataAwal);
46
47  //fungsi cari nilai min
48  public void seekingMin(double[][] value) {
49
50      minValue = new double[value[0].length];
51
52      for (int i = 0; i < value[0].length; i++) {
53          double temp = value[0][0];
54          for (int k = 0; k < value.length; k++) {
55              temp = cekMin(temp, value[k][i]);
56          }
57          this.minValue[i] = temp;
58      }
59  }
60
61  //fungsi cek nilai min untuk perbandingan pada perulangan
62      public double cekMin(double value1, double value2) {
63
64          if (value1 < value2) {
65              return value1;
66          } else {
```

```
67          return value2;
68        }
69     }
70
71
72  // Generate dataNormalisasi
73  for (int i = 0; i < dataNormalisasi.length; i++) {
74     for (int k = 0; k < dataNormalisasi[0].length; k++) {
75        dataNormalisasi[i][k]=normalization(dataAwal[i][k],
76  this.maxValue[k], this.minValue[k]);
77     }
78  }
79
```

Gambar 5.1 Implementasi Normalisasi Data

- Baris 1-5 inisialisasi data awal dan data normalisasi.
- Baris 9-13 perulangan i untuk memasukkan data awal sepanjang jumlah kolom, perulangan K untuk memasukkan data awal sepanjang jumlah baris.
- Baris 16-31 mencari nilai maksimum dengan menggunakan fungsi seekingMax, pencarian nilai max menggunakan perulangan dengan memanggil fungsi cekMax.
- Baris 36-40 kondisi untuk menemukan nilai yang lebih besar dari nilai yang akan dibandingkan.
- Baris 48-59 mencari nilai minimum dengan menggunakan fungsi seekingMin, pencarian nilai min menggunakan perulangan dengan memanggil fungsi cekMin.
- Baris 64-67 kondisi untuk menemukan nilai yang lebih kecil dari nilai yang akan dibandingkan.
- Baris 73-76 Perulangan i untuk memasukkan datanormalisasi sepanjang jumlah kolom, perulangan k untuk memasukkan data normalisasi sepanjang jumlah baris.

5.1.2 Implementasi Proses inisialisasi input *Weight* dan *Bias*

pada tahap implementasi proses insisalisasi *input weight* dan *bias* ini menggunakan nilai *input weight* dan *bias* secara random dengan *range* nilai sebesar 1 sampai -1. Jumlah dari *input weight* dan *bias* tergantung dari jumlah *neuron* yang telah dimasukkan oleh *user* kedalam sistem. Gambar 5.2 merupakan implementasi proses inisialisasi *input weight* dan *bias.*

```
1   double[][] weightValue1, weightValue2, weightValue3;
2    double[] bias = new double[2];
3   int max = 1;
4   int min = -1;
```

```
5
6     for (int i=0; i < hiddenNeuron; i++)
7        {
8          this.bias[i] = Math.random()*(max-min)+min;
9        }
10  weightValue1 = new double[variasi1][bias.length];
11    for (int i = 0; i < weightValue1.length; i++) {
12       for (int k = 0; k < weightValue1[0].length; k++) {
13          weightValue1[i][k] = Math.random()*(max-min)+min;
14    }
15  }
16  weightValue2 = new double[variasi2][bias.length];
17    for (int i = 0; i < weightValue2.length; i++) {
18        for (int k = 0; k < weightValue2[0].length; k++) {
19          weightValue2[i][k] = Math.random()*(max-min)+min;;
20    }
21  }
22  weightValue3 = new double[variasi3][bias.length];
23    for (int i = 0; i < weightValue3.length; i++) {
24       for (int k = 0; k < weightValue3[0].length; k++) {
25          weightValue3[i][k] = Math.random()*(max-min)+min;;
26     }}
27
```

Gambar 5.2 Implementasi Inisialisai *Weight* dan *Bias*

- baris 1-4 : inisialisasi variabel weightValue1, weightValue2, weightValue3, bias, min, dan max.
- Baris 6-9 : perulangan untuk mencari nilai bias secara random. Perulangan dilakukan sebanyak jumlah *neuron* pada *hidden layer.*
- Baris 10-27 : rumus perulangan untuk mencari nilai dari *input weight*. Perulangan dilakukan sesuai dengan jumlah *neuron* pada *hidden layer* dan variasi fitur.

5.1.3 Implementasi Proses Hitung Keluaran *Hidden Layer*

pada implementasi proses hitung keluaran *hidden layer* ini merupakan perhitungan antara data yang telah dinormalisasi dengan *input weight* dan *bias.* Proses ini digunakan baik pada saat proses *training* dan *testing*. Gambar 5.3 merupakan implementasi proses hitung keluaran *hidden layer.*

```
1  // Generate hInitTraining
2  for (int i = 0; i < hInit.length; i++) {
3     for (int k = 0; k < hInit[0].length; k++) {
4  this.hInit[i][k] = initFunction(this.weight[k],
5  dataNormalisasi[i], this.bias[k]);
6  public double initFunction(double[] value1, double[]
7  value2, double bias) {
8     double temp = 0.0;
9     for (int i = 0; i < value1.length; i++) {
10       temp += value1[i] * value2[i];
11    }
12    temp += bias;
13    return temp;
14 }
15 // Generate hInitTesting
16 for (int i = 0; i < hInitTesting.length; i++) {
17    for (int k = 0; k < hInitTesting[0].length; k++) {
18 hInitTesting[i][k] = initFunction(this.weight[k],
19 dataNormalisasi[hInit.length + i], this.bias[k]);
20    }
21 }
22 public double initFunction(double[] value1, double[] value2, double bias)
23 {
24    double temp = 0.0;
25    for (int i = 0; i < value1.length; i++) {
26       temp += value1[i] * value2[i];
27    }
```

```
28      temp += bias;
29      return temp;
        }
```

Gambar 5.3 Implementasi Proses Hitung Keluaran *Hidden Layer*

- Baris 2-4 : perulangan untuk mencari nilai HinitTraining dengan menggunakan fungsi initFunction dengan parameter nilai dari *weight,* datanormalisasi, dan bias.
- Baris 6-14 : merupakan fungsi initFunction yang berisi perhitungan perkalian nilai data normalisasi dengan *input weight* dan *bias*.
- Baris 16-21 : Perulangan untuk mencari nilai HinitTraining dengan menggunakan fungsi initFunction dengan parameter nilai *weight,* data normalisasi, dan bias.
- Baris 22-29 : merupakan fungsi initFunction yang berisi perhitungan perkalian nilai data normalisasi dengan *input weight* dan *bias.*

5.1.4 Implementasi Proses Hitung Keluaran *Hidden Layer* Dengan Fungsi Aktivasi

pada implementasi proses hitung keluaran *hidden layer* dengan fungsi aktivasi ini merupakan perhitungan keluaran *hidden layer* yang diproses dengan fungsi aktivasi sigmoid. Proses ini digunakan baik pada saat proses *training* dan *testing*. Gambar 5.4 merupakan implementasi proses hitung keluaran *hidden layer* dengan fungsi aktivasi.

```
1   // Generate hActivation
2   for (int i = 0; i < hActivation.length; i++) {
3      for (int k = 0; k < hActivation[0].length; k++) {
4         hActivation[i][k] = activationFunction(hInit[i][k]);
5      }
6   }
7   public double activationFunction(double value) {
8       return 1 / (1 + Math.pow(E, -value));
9   }
```

Gambar 5.4 Implementasi Proses Hitung Keluaran *Hidden Layer* Dengan Fungsi Aktivasi

- Baris 2-6 : perulangan untuk mencari nilai Hinit menggunakan fungsi activationFunction.
- Baris 7-9 : merupakan implementasi dari fungsi aktivasi sigmoid.

5.1.5 Implementasi Proses Hitung *Output Weight*

pada implementasi proses hitung *output weight* ini adalah mengolah data *training* dengan melalui beberapa proses seperti yang sudah dipaparkan pada bab 4. Gambar 5.5 merupakan implementasi proses hitung *output weight.*

```
1   // Transpose hActivation
2   this.hActivationTranspose = transposeMatrix(hActivation);
3   public double[][] transposeMatrix(double[][] value) {
4   double[][]temp = new double[value[0].length][value.length];
5   for (int i = 0; i < value[0].length; i++) {
6      for (int k = 0; k < value.length; k++) {
7         temp[i][k] = value[k][i];
8         }
9      }
10     return temp;
11  }
12  this.hResult = matrixProduct(hActivationTranspose,
13  hActivation);
14  public double[][] matrixProduct(double[][] value1, double[][]
15  value2) {
16    double[][]temp=new double[value1.length][value2[0].length];
17    for (int i = 0; i < temp.length; i++) {
18       for (int k = 0; k < temp[0].length; k++) {
19          temp[i][k] = 0;
20          for (int m = 0; m < value1[0].length; m++) {
21              temp[i][k] += value1[i][m] * value2[m][k];
22          }
23       }
24    }
25    return temp;
26  }
27  this.hResultInverse = inverseMatrix2by2(hResult);
28
```

```
public double[][] inverseMatrix2by2(double[][] value) {
    double[][] temp = new double[2][2];
    double determinance = (value[0][0] * value[1][1]) - (value[0][1] *
value[1][0]);
    temp[0][0] = 1 / determinance * value[1][1];
    temp[0][1] = 1 / determinance * -value[0][1];
    temp[1][0] = 1 / determinance * -value[1][0];
    temp[1][1] = 1 / determinance * value[0][0];
    return temp;
}
this.hPlus = matrixProduct(hResultInverse,
hActivationTranspose);
public double[][] matrixProduct(double[][] value1,
double[][] value2) {
   double[][]temp=newdouble[value1.length][value2[0].length];
   for (int i = 0; i < temp.length; i++) {
     for (int k = 0; k < temp[0].length; k++) {
       temp[i][k] = 0;
       for (int m = 0; m < value1[0].length; m++) {
          temp[i][k] += value1[i][m] * value2[m][k];
       }
     }
   }
   return temp;
}
this.beta = matrixProduct2(hPlus, dataNormalisasi);
public double[] matrixProduct2(double[][] value1,
double[]value2) {
   double[] temp = new double[value1.length];
   for (int i = 0; i < temp.length; i++) {
     temp[i] = 0;
```

```
60      for (int m = 0; m < value1[0].length; m++) {
61    temp[i] += value1[i][m] * value2[m] [this.variation];
62      }
63    }
64      return temp;
    }
```

Gambar 5.5 Implementasi Proses Hitung *Output Weight*

- Baris 2 : menjalankan fungsi transposeMatrix, menggunakan nilai hActivation yang sudah diperoleh pada proses sebelumnya.
- Baris 3-11 : merupakan fungsi transposeMatrikx yang berisi perulangan untuk mendapatkan hActivationTranspose.
- Baris 12-13 : menjalankan fungsi matrixProduct, menggunakan parameter hActivationTranspose dan hActivation.
- Baris 14-26 : merupakan fungsi matrixProduct yang digunakan untuk mencari nilai dari hResult yang didapatkan dengan mengalikan hActivationTranspose dengan hActivation.
- Baris 27 : menjalankan fungsi inverseMatrix2by2, menggunakan parameter hResult.
- Baris 28-37 : merupakan fungsi inverseMatrix2by2, didalamnya terdapat proses inverse matriks hResult.
- Baris 38-39 : menjalankan fungsi matrixProduct, menggunakan parameter hResultInverse dan hActivationTranspose.
- Baris 40-52 : merupakan fungsi matrixProduct yang digunakan untuk mencari nilai dari hPlus yang didapatkan dengan mengalikan hResultInverse dan hActivationTranspose.
- Baris 53 : menjalankan fungsi matrixProduct2, menggunakan parameter hPlus dan dataNormalisasi untuk menghasilkan *output weight.*
- Baris 54-64 : merupakan fungsi matrixProduct2 yang digunakan untuk mencari nilai beta dengan mengkalikan hPlus dengan dataNormalisasi dari Target yang telah diperoleh.

5.1.6 Implementasi Proses Hitung Keluaran di *Output Layer*

Pada implementasi proses hitung *output layer* ini adalah mengolah data *testing* dengan melalui beberapa proses seperti yang sudah dipaparkan pada bab 4. Gambar 5.6 merupakan implementasi proses hitung *output layer.*

```
1   this.hInitTesting = new double[2][bias.length];
2   this.hActivTesting = new double[2][bias.length];
3   // Generate hInitTesting
4   for (int i = 0; i < hInitTesting.length; i++) {
```

```
5     for (int k = 0; k < hInitTesting[0].length; k++) {
6     hInitTesting[i][k] = initFunction(this.weight[k],
7  dataNormalisasi[hInit.length + i], this.bias[k]);
8           }
9        }
10 public double initFunction(double[] value1, double[] value2, double bias) {
11   double temp = 0.0;
12   for (int i = 0; i < value1.length; i++) {
13      temp += value1[i] * value2[i];
14   }
15   temp += bias;
16   return temp;
17 }
18 // Generate hActivTesting
19 for (int i = 0; i < hActivTesting.length; i++) {
20    for (int k = 0; k < hActivTesting[0].length; k++) {
21       hActivTesting[i][k] =
22 activationFunction(hInitTesting[i][k]);
23          }
24       }
25 public double activationFunction(double value) {
26    return 1 / (1 + Math.pow(E, -value));
27 }
28 // Generates way
29 this.way = matrixProduct2(hActivTesting, beta);
30 public double[] matrixProduct2(double[][] value1,
31 double[]value2) {
32    double[] temp = new double[value1.length];
33    for (int i = 0; i < temp.length; i++) {
34       temp[i] = 0;
35       for (int m = 0; m < value1[0].length; m++) {
```

```
36      temp[i] += value1[i][m] * value2[m];
37          }
38      }
39          return temp;
40  }
41
```

Gambar 5.6 Implementasi Proses Hitung *Output layer*

- Baris 1-2 : inisialisasi hInintTesting dan hActivTesting.
- Baris 4-9 : perulangan untuk mencari nilai HinitTesting dengan menggunakan fungsi initFunction dengan parameter nilai dari *weight,* datanormalisasi, dan bias.
- Baris 10-18 : merupakan fungsi initFunction yang berisi perhitungan nilai data normalisasi dengan *input weight* dan *bias*.
- Baris 20-25: perulangan untuk mencari nilai hActiveTesting menggunakan fungsi activationFunction dengan nilai HinitTesting.
- Baris 26-28 : merupakan implementasi dari fungsi aktivasi sigmoid.
- Baris 30 : menjalankan fungsi matrixProduct2, menggunakan parameter nilai hActivTesting dan beta.
- Baris 31-41 : merupakan fungsi matrixProduct2 yang digunakan untuk mencari nilai way dengan mengalikan beta dan hActivTesting.

5.1.7 Implementasi Proses Hitung nilai *Mean Square Error* (MSE)

Pada implementasi proses hitung nilai *mean square error* (MSE) ini adalah mengolah data way yang mana nilai error yang diapatkan tersebut berasal dari membandingkan target yang ternormalisasi dengan way sesuai dengan proses seperti yang sudah dipaparkan pada bab 4. Gambar 5.7 merupakan implementasi proses hitung nilai *mean square error* (MSE).

```
1   double[] tempForMSE = new double[way.length];
2   for (int i = 0; i < way.length; i++){
3      tempForMSE[i] = way[i] - dataNormalisasi[hInit.length +
4   i][this.variation - 1];
5      }
6   this.mseValue = msefunction(tempForMSE);
7   public double msefunction(double[] value) {
8     double temp = 0.0;
9     for (int i = 0; i < value.length; i++) {
```

```
10        temp += Math.pow(value[i], 2);
11      }
12      return temp / value.length;
13    }
14  }
```

Gambar 5.7 Implementasi Proses Hitung Nilai *Mean Square Error* (MSE)

- Baris 1 : inisialisasi tempForMSE untuk nilai sementara perhitungan MSE.
- Baris 2-5 : perulangan untuk mencari nilai tempForMSE dengan pengurangan nilai way dan nilai normalisasi.
- Baris 6 : menjalankan fungsi msefunction dengan parameter nilai tempForMSE.
- Baris 7-13 : fungsi untuk mencari nilai MSE dengan cara penjumlahan semua hasil kuadrat tempForMSE dibagi dengan jumlah data tempForMSE.

5.1.8 Implementasi Proses Denormalisasi Data

pada implementasi proses denormalisasi data ini merubah data way menjadi data yang normal. Gambar 5.8 merupakan implementasi proses normalisasi data.

```
1   this.dataDenormalisasi = new double[way.length];
2   for (int i = 0; i < way.length; i++){
3     dataDenormalisasi[i] = denormalization(way[i],
4   this.maxValue[this.maxValue.length - 1],
5   this.minValue[this.minValue.length - 1]);
6   }
7   public double denormalization(double value, double max,
8   double min) {
9     return (value * (max - min)) + min;
10  }
```

Gambar 5.8 Implementasi Proses Denormalisasi Data

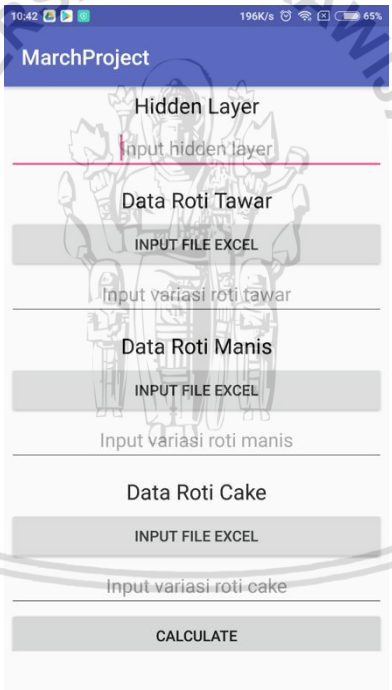
- Baris 1 : inisialisasi dataDenormalisasi
- Baris 2-6 : mengisi data denormalisasi dengan menjalankan fungsi denormalization menggunakan parameter way, maxValue, dan minValue
- Baris 7-10 : merupakan fungsi denormalization untuk mencari nilai dari perkalian way dengan selisih maxValue dan minValue ditambah minValue.

5.2 Imlementasi Antarmuka

pada subab implementasi antarmuka ini, berisi tentang *screenshoot* dari program yang telah dibuat, yang mana pada penelitian ini peneliti menggunakan implementasi berbasis android sesuai dengan perancangan dan pengujian pada bab 4. Aplikasi ini memiliki beberapa halaman yaitu halaman *import* data, normalisasi, *training, testing,* denormalisasi, evaluasi, dan halaman *weight & bias.*

5.2.1 Implementasi Halaman *Import* Data

Pada Implementasi *import* data ini berfungsi sebagai halaman utama dari aplikasi, pada halaman ini pengguna harus memasukkan beberapa data seperti data historis penjualan roti cake, manis, dan tawar. Pengguna juga diminta untuk mengisi variasi atau fitur data historis penjualan ketiga jenis roti tersebut, dan kemudian pengguna diminta untuk mengisi jumlah *hidden neuron*. Implementasi halaman *import* data diperlihatkan pada Gambar 5.9.

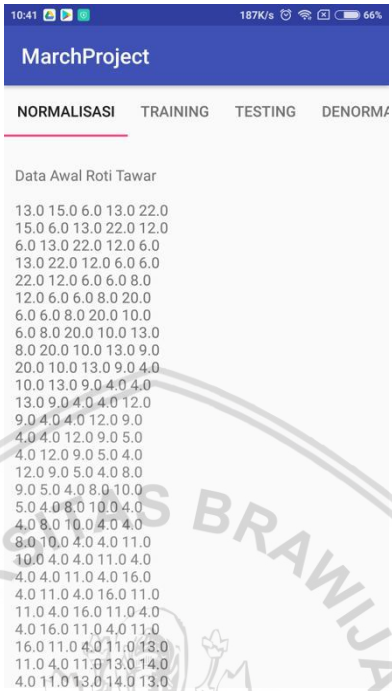


Gambar 5.9 Implementasi Halaman *Import* Data

5.2.2 Implementasi Halaman Normalisasi

Pada implementasi halaman normlaisasi ini berfungsi sebagai halaman yang menampilkan hasil dari normalisasi data awal yang telah dimasukkan oleh user yang diantaranya adalah data historis penjualan roti manis, roti tawar, dan cake. Data yang digunakan pada normalisasi ini adalah sebesar 91 data selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang berisis data penjualan roti pada Harum *Bakery* dengan menggunakan 4 variasi fitur, menggunakan 2

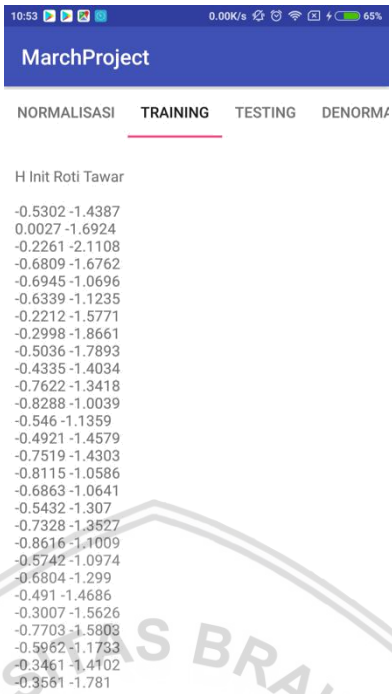
neuron pada *hidden layer*, dan presentase data *training* sebesar 80% dan data *testing* sebesar 20%. Implementasi halaman normalisasi diperlihatkan pada Gambar 5.10.



Gambar 5.10 Implementasi Halaman Normalisasi

5.2.3 Implementasi Halaman *Training*

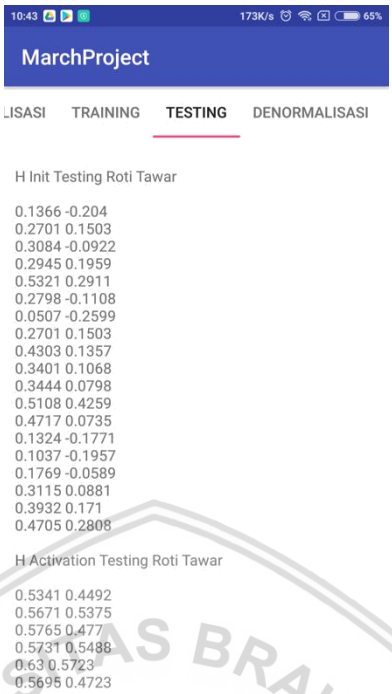
Pada implementasi halaman *training* ini berfungsi sebagai halaman yang menampilkan hasil dari proses data *training* yang mana pada prosesnya terdapat beberapa langkah yakni, menghitung keluaran *hidden layer*, kemudian keluaran tersebut dihitung dengan fungsi aktivasi sigmoid, dan *output weight* Data yang digunakan pada *training* ini adalah sebesar 91 data selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang berisis data penjualan roti pada Harum *Bakery* yang telah dinormalisasi dengan menggunakan 4 variasi fitur, menggunakan 2 neuron pada *hidden layer*, dan presentase data *training* yang digunakan sebesar 80% dari keseluruhan data. Implementasi halaman *training* akan diperlihatkan pada Gambar 5.11.



Gambar 5.11 Implementasi Halaman *Training*

5.2.4 Implementasi Halaman *Testing*

Pada implementasi halaman *testing* ini berfungsi sebagai halaman yang menampilkan hasil dari proses data *testing* yang mana prosesnya terdapat beberapa langkah yakni, menhitung keluaran *hidden layer,* keluaran *hidden layer* yang diproses dengan fungsi aktivasi sigmoid, dan kemudian mengitung *output layer* yang merupakan proses lanjutan dari *output weight* yang telah didapatkan dari proses *training*. Data yang digunakan adalah data historis penjualan roti manis, roti tawar, dan roti cake selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang masing-masing memiliki total 91 data per jenis rotinya yang telah dinormalisasi dengan menggunakan 4 variasi fitur, menggunakan 2 neuron pada *hidden layer*, dan presentase data *testing* yang digunakan sebesar 20% dari keseluruhan data. Implementasi halaman *testing* akan diperlihatkan pada Gambar 5.12.



Gambar 5.12 Implementasi Halaman *Testing*

5.2.5 Implementasi Halaman Denormalisasi

Pada implementasi halaman denormalisasi ini berfungsi sebagai halaman yang menunjukkan proses denormalisasi hasil prediksi yang didapatkan dari proses *testing*. Implementasi halaman denormalisasi akan diperlihatkan pada Gambar 5.13

Gambar 5.13 Implementasi Halaman Denormalisasi

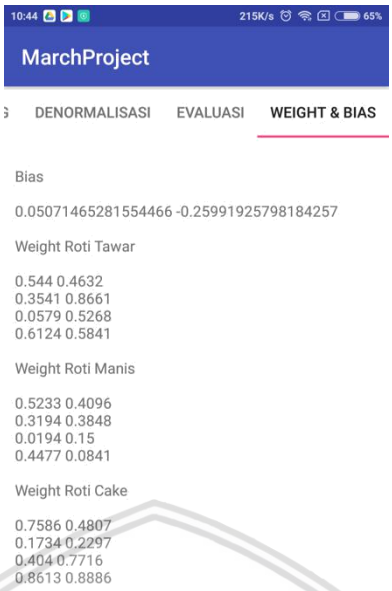
5.2.6 Implementasi Halaman Evaluasi

Pada implementasi halaman evaluasi ini berfungsi sebagai halaman yang menunjukkan proses evaluasi hasil prediksi berupa nilai erornya dimana pada prediksian ini menggunakan *Mean Square Error(*MSE*)* sebagai metode evaluasinya. Implementasi halaman evaluasi akan diperlihatkan pada Gambar 5.14

Gambar 5.14 Implementasi Halaman Evaluasi

5.2.7 Implementasi Halaman *Weight & Bias*

Pada implementasi halaman *weight & bias* ini berfungsi sebagai halaman yang menunjukkan nilai dari *input weight* dan *bias* yang didapatkan secara random oleh sistem. Implementasi halaman *weight & bias* akan diperlihatkan pada Gambar 5.15



Gambar 5.14 Implementasi Halaman *Weight & Bias*

BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini akan menjelaskan pengujian dan pembahasan dari hasil perhitungann jumlah penjualan roti dengan menggunakan metode *Extreme Learning Machine*. Pengujian ini pada dasarnya adalah mencari rata-rata nilai MSE terkecil atau yang terbaik yang kemudian peneliti menggunakan 3 pengujian yang akan diuji dalam penelitian ini, yaitu pengujian jumlah *neuron* pada *hidden layer*, pegujian jumlah data historis penjualan, dan pengujian jumlah fitur data historis penjualan. Setiap pengujian dilakukan 10 kali percobaan dan kemudian hasil dari 10 kali percoban tersebut dicari rata-ratanya MSE.

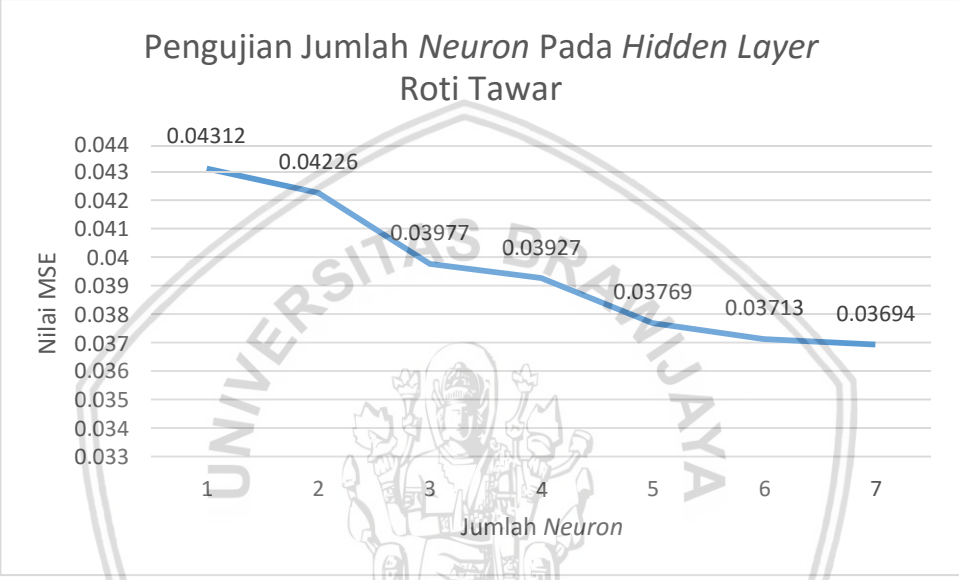
6.1 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Pengujian ini dilakukan untuk mendapatkan nilai MSE yang paling rendah atau yang terkecil. Pengujian ini adalah pengujian jumlah *neuron* pada *hidden layer.* Pengujian ini menguji jumlah *neuron* pada *hidden layer* mulai dari 1 *neuron* sampai 7 *neuron*. Data yang digunakan adalah data historis penjualan roti manis, roti tawar, dan roti cake selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang masing-masing memiliki total 91 data per jenis rotinya. Peneitian ini menggunakan 4 fitur untuk pengujiannya. Makusd dari 4 fitur ini adalah fitur data historis penjualan selama 4 hari keblakang. Jadi pengujian ini menggunakan data hari h-4 sampai hari h-1 hari sebelum target prediksi. Pengujian ini juga menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya. Perbandingan data *training* dan data *testing* adalah 80% dan 20%. Setiap pengujian jumlah *neuron* pada *hidden layer* dilakukan sebanyak 10 kali. Setiap kali melakukan percobaan, sistem akan menampilkan nilai MSE nya, kemudian dari 10 kali percobaan itu nilai MSE nya dirata-rata dan kemudian nilai rata-rata MSE tersebut di bandingkan dengan pengujian jumlah *neuron* pada *hidden neuron* yang lain. Hasil pengujian dari pengujian jumlah *neuron* pada *hidden layer* untuk masing-masing roti akan ditunjukkan pada Tabel 6.1, Tabel 6.2, dan Tabel 6.3. Sedangkan untuk grafik hasil pengujian jumlah *neuron* pada *hidden layer* ditunjukkan pada Gambar 6.1, Gambar 6.2, dan Gambar 6.3.

Tabel 6.1 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Tawar

Percobaan ke-*i*	Nilai MSE pada Jumlah Hidden Neuron						
	1	2	3	4	5	6	7
1	0.0365	0.0425	0.0385	0.0395	0.0362	0.0365	0.0328
2	0.0457	0.0375	0.0344	0.0363	0.0372	0.039	0.0379
3	0.0564	0.043	0.0373	0.033	0.0389	0.0366	0.0326
4	0.04	0.0349	0.0394	0.0493	0.0369	0.0358	0.0397
5	0.0429	0.0488	0.0449	0.0411	0.0372	0.0363	0.0336

6	0.0325	0.0551	0.0459	0.035	0.0364	0.0357	0.0383
7	0.0405	0.0313	0.0351	0.047	0.0382	0.0395	0.0356
8	0.0439	0.0519	0.0358	0.0419	0.0412	0.0354	0.0451
9	0.0413	0.0384	0.0508	0.0364	0.0365	0.0357	0.0385
10	0.0515	0.0392	0.0356	0.0332	0.0382	0.0408	0.0353
Rata-Rata nilai MSE	0.04312	0.04226	0.03977	0.03927	0.03769	0.03713	0.03694



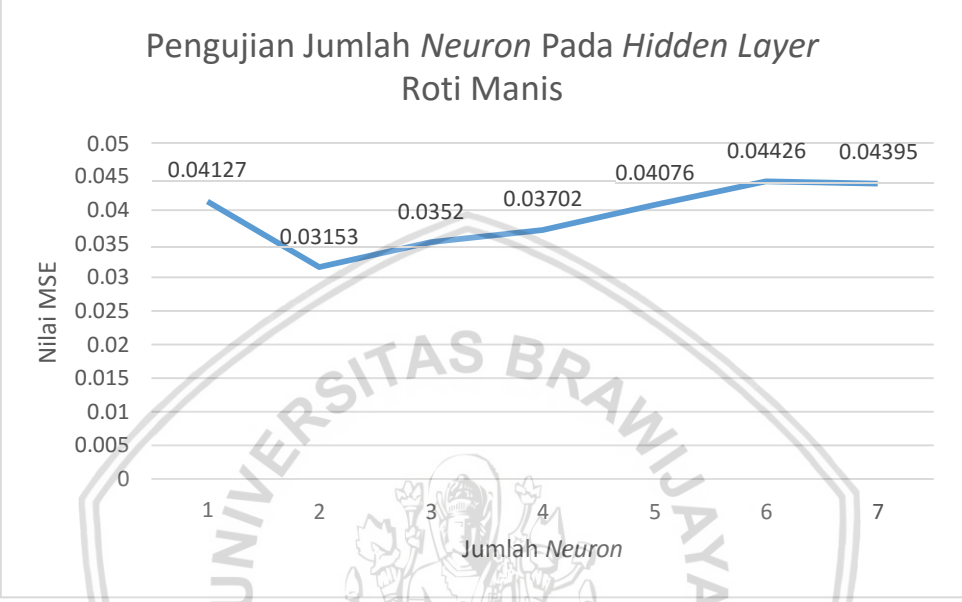
Gambar 6.1 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Tawar

Sesuai dengan tabel 6.1 dan grafik pada gambar 6.1, nilai MSE terbaik pada roti tawar dihasilkan dengan menggunakan 7 neuron pada *hidden layer* dengan MSE sebesar 0,0369.

Tabel 6.2 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Manis

Percobaan ke-*i*	**Nilai MSE pada Jumlah Hidden Neuron**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0.0421	0.0267	0.0367	0.0384	0.0432	0.0469	0.0437
2	0.0404	0.0214	0.0363	0.0358	0.0396	0.0423	0.0458
3	0.0341	0.0371	0.0324	0.0386	0.0413	0.0428	0.0439
4	0.0446	0.037	0.0306	0.0406	0.043	0.0409	0.0399
5	0.0349	0.0324	0.0323	0.0295	0.042	0.0454	0.0429
6	0.0375	0.0288	0.038	0.037	0.0374	0.0405	0.0454
7	0.0337	0.03	0.0315	0.0362	0.044	0.0434	0.0426

8	0.0383	0.0309	0.0414	0.042	0.0423	0.0427	0.0454
9	0.0541	0.0371	0.0415	0.0364	0.0385	0.0539	0.0447
10	0.053	0.0339	0.0313	0.0357	0.0363	0.0438	0.0452
Rata-Rata nilai MSE	0.04127	0.03153	0.0352	0.03702	0.04076	0.04426	0.04395



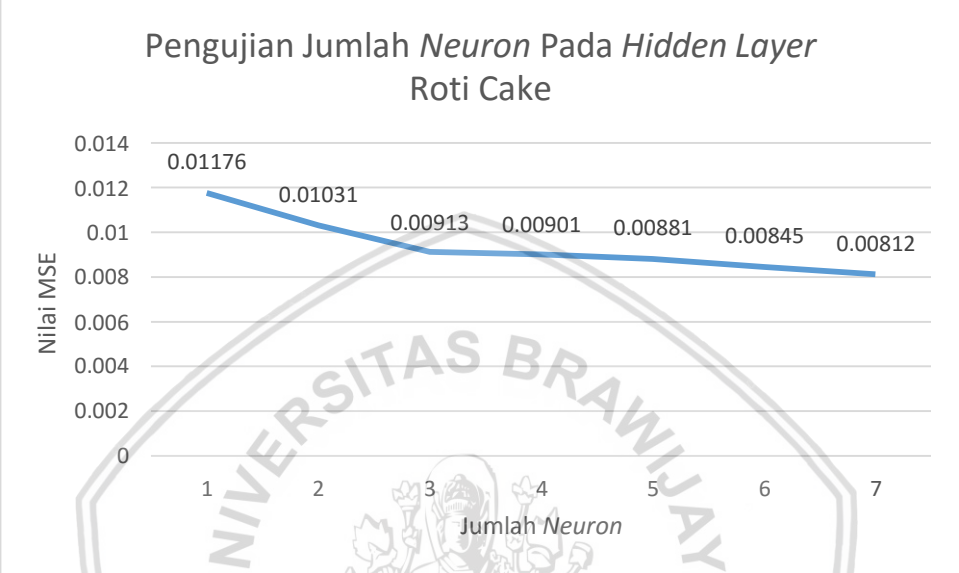
Gambar 6.2 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Manis

Sesuai dengan tabel 6.2 dan grafik pada gambar 6.2, nilai MSE terbaik pada roti manis dihasilkan dengan menggunakan 2 neuron pada *hidden layer* dengan MSE sebesar 0,03153.

Tabel 6.3 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Cake

Percobaan ke-*i*	**Nilai MSE pada Jumlah Hidden Neuron**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0.0108	0.0124	0.0088	0.0094	0.0085	0.0084	0.0085
2	0.011	0.0094	0.007	0.0086	0.0106	0.0082	0.008
3	0.0123	0.0094	0.0086	0.0088	0.0085	0.0083	0.0083
4	0.0128	0.011	0.0099	0.0083	0.0088	0.0091	0.0077
5	0.0127	0.0109	0.0097	0.0091	0.0084	0.0077	0.0082
6	0.0106	0.0103	0.0106	0.0092	0.0084	0.0083	0.0074
7	0.0106	0.0103	0.0103	0.0082	0.0086	0.0084	0.0081

8	0.0128	0.0098	0.0103	0.0117	0.0093	0.0088	0.0087
9	0.0113	0.009	0.0078	0.0083	0.0085	0.0088	0.0084
10	0.0127	0.0106	0.0083	0.0085	0.0085	0.0085	0.0079
Rata-Rata nilai MSE	0.01176	0.01031	0.00913	0.00901	0.00881	0.00845	0.00812



Gambar 6.3 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer* Roti Cake

Sesuai dengan tabel 6.3 dan grafik pada gambar 6.3, nilai MSE terbaik pada roti cake dihasilkan dengan menggunakan 7 neuron pada *hidden layer* dengan MSE sebesar 0,00812. Dengan pengujian yang telah dilakukan, dapat disimpulkan bahwa semakin banyak jumlah neuron tidak menjamin membaiknya nilai MSE, dikarenakan data yang memiliki nilai variasi yang tinggi.

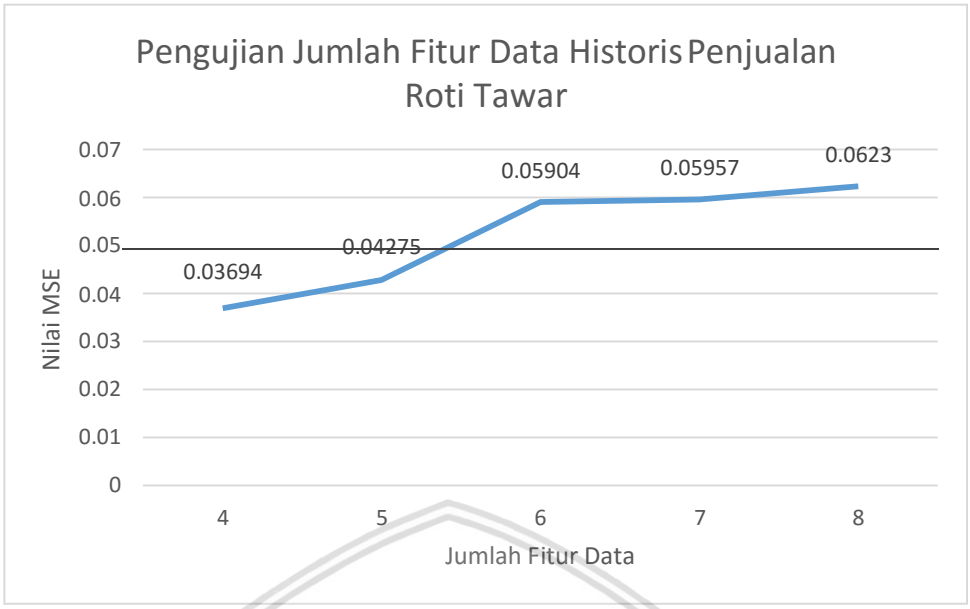
6.2 Pengujian Jumlah Fitur Data Historis Penjualan

Pengujian ini dilakukan untuk mendapatkan nilai MSE yang paling rendah atau yang terkecil. Pengujian ini adalah pengujian jumlah fitur data historis penjualan. Pengujian ini menguji jumlah fitur data historis penjualan mulai dari 4 sampai 8 fitur data historis penjualan. Data yang digunakan adalah data historis penjualan roti manis, roti tawar, dan roti cake selama 3 bulan mulai dari bulan September 2017-Nopember 2017 yang masing-masing memiliki total 91 data per jenis rotinya. Jumlah *neouron* pada *hidden layer* pada pengujian ini menggunakan jumlah *neuron* yang memiliki nilai MSE terkecil pada pengujian jumlah *neuron* pada *hidden layer* sebelumnya yaitu menggunakan 7 *neuron* untuk roti tawar dan cake, dan menggunakan 2 *neuron* pada *hiden layer* untuk roti manis. Pengujian ini menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya. Perbandingan data *training* dan data *testing* adalah 80% dan 20%. Setiap pengujian jumlah fitur data historis penjualan dilakukan sebanyak 10 kali.

Setiap kali melakukan percobaan, sistem akan menampilkan nilai MSE nya, kemudian dari 10 kali percobaan itu nilai MSE nya dirata-rata dan kemudian nilai rata-rata MSE tersebut di bandingkan dengan pengujian jumlah fitur data yang lain. Hasil pengujian dari pengujian jumlah fitur data historis penjualan untuk masing-masing roti akan ditunjukkan pada Tabel 6.4, Tabel 6.5, dan Tabel 6.6. Sedangkan untuk grafik hasil pengujian jumlah fitur data historis penjualan ditunjukkan pada Gambar 6.4, Gambar 6.5, dan Gambar 6.6.

Tabel 6.4 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Tawar

Percobaan ke-*i*	**Nilai MSE pada Jumlah Fitur Data Historis Penjualan**				
	4	**5**	**6**	**7**	**8**
1	0.0328	0.0414	0.0587	0.0557	0.0612
2	0.0379	0.041	0.0583	0.0709	0.0635
3	0.0326	0.0467	0.0605	0.0626	0.0647
4	0.0397	0.0385	0.0557	0.064	0.0637
5	0.0336	0.0509	0.0606	0.046	0.0638
6	0.0383	0.0461	0.0585	0.0576	0.0695
7	0.0356	0.0419	0.059	0.057	0.0708
8	0.0451	0.0391	0.0593	0.0573	0.0591
9	0.0385	0.039	0.059	0.0633	0.0547
10	0.0353	0.0429	0.0608	0.0613	0.052
Rata-rata Nilai MSE	0.03694	0.04275	0.05904	0.05957	0.0623

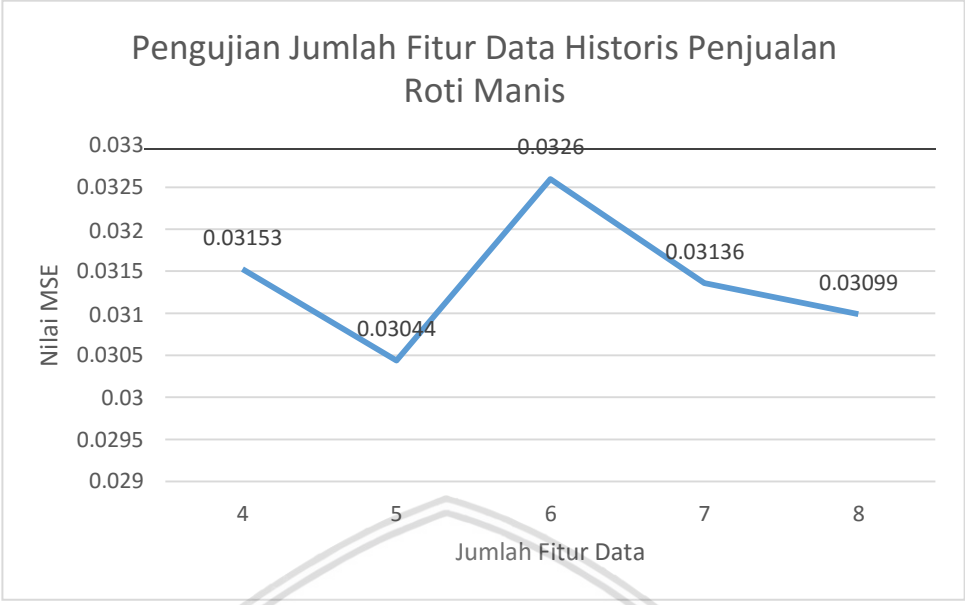


Gambar 6.4 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Tawar

Sesuai dengan tabel 6.4 dan grafik pada gambar 6.4, nilai MSE terbaik pada roti tawar dihasilkan dengan menggunakan 4 fitur dengan MSE sebesar 0,03694.

Tabel 6.5 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Manis

Percobaan ke-*i*	**Nilai MSE pada Jumlah Fitur Data Historis Penjualan**				
	4	**5**	**6**	**7**	**8**
1	0.0267	0.027	0.035	0.0325	0.0251
2	0.0214	0.028	0.0374	0.0361	0.0395
3	0.0371	0.0313	0.0345	0.0291	0.029
4	0.037	0.0337	0.0396	0.0311	0.0263
5	0.0324	0.0288	0.0257	0.0318	0.0256
6	0.0288	0.0319	0.035	0.0401	0.0371
7	0.03	0.0304	0.037	0.0243	0.0348
8	0.0309	0.0323	0.0312	0.0288	0.0356
9	0.0371	0.0252	0.021	0.027	0.0254
10	0.0339	0.0358	0.0296	0.0328	0.0315
Rata-rata Nilai MSE	0.03153	0.03044	0.0326	0.03136	0.03099

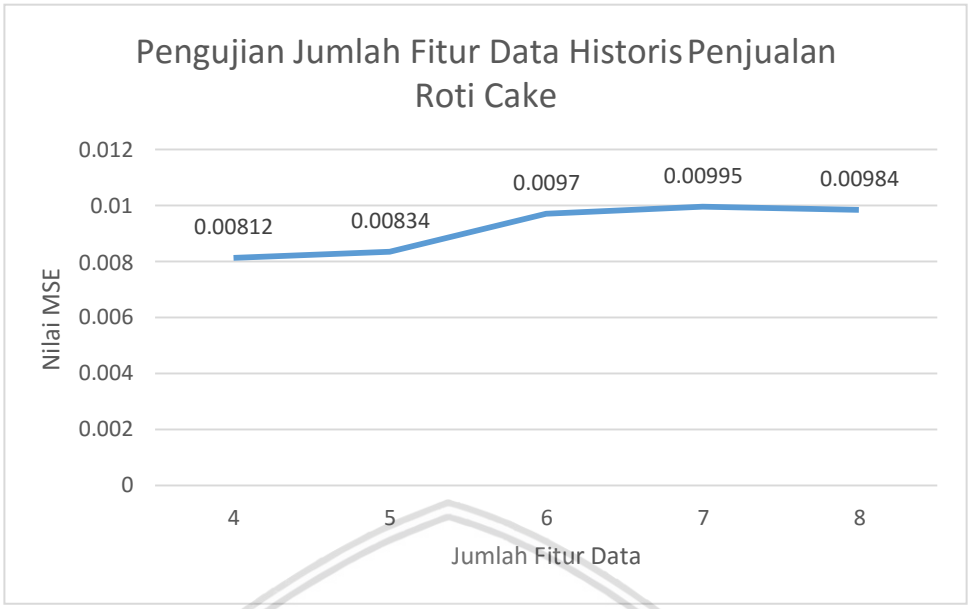


Gambar 6.5 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Manis

Sesuai dengan tabel 6.5 dan grafik pada gambar 6.5, nilai MSE terbaik pada roti manis dihasilkan dengan menggunakan 5 fitur dengan MSE sebesar 0,03044.

Tabel 6.6 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Cake

Percobaan ke-*i*	**Nilai MSE pada Jumlah Fitur Data Historis Penjualan**				
	4	**5**	**6**	**7**	**8**
1	0.0085	0.0079	0.0098	0.0084	0.0106
2	0.008	0.0084	0.0101	0.0105	0.0098
3	0.0083	0.0077	0.0083	0.011	0.0103
4	0.0077	0.0085	0.011	0.0107	0.0096
5	0.0082	0.0074	0.0096	0.0103	0.0092
6	0.0074	0.008	0.0099	0.0089	0.0083
7	0.0081	0.0117	0.0094	0.0099	0.0096
8	0.0087	0.0078	0.0088	0.0102	0.0113
9	0.0084	0.008	0.0103	0.0094	0.0087
10	0.0079	0.008	0.0098	0.0102	0.011
Rata-rata Nilai MSE	0.00812	0.00834	0.0097	0.00995	0.00984



Gambar 6.6 Hasil Pengujian Jumlah Fitur Data Historis Penjualan Roti Cake

Sesuai dengan tabel 6.6 dan grafik pada gambar 6.6, nilai MSE terbaik pada roti cake dihasilkan dengan menggunakan 4 fitur dengan MSE sebesar 0,00812. Dapat disimpulkan bahwa semakin banyak fitur yang digunakan, maka data latih akan semakin sedikit, sedangkan semakin banyak jumlah data training dan testing, maka semakin baik pula tingkat prediksiannya.

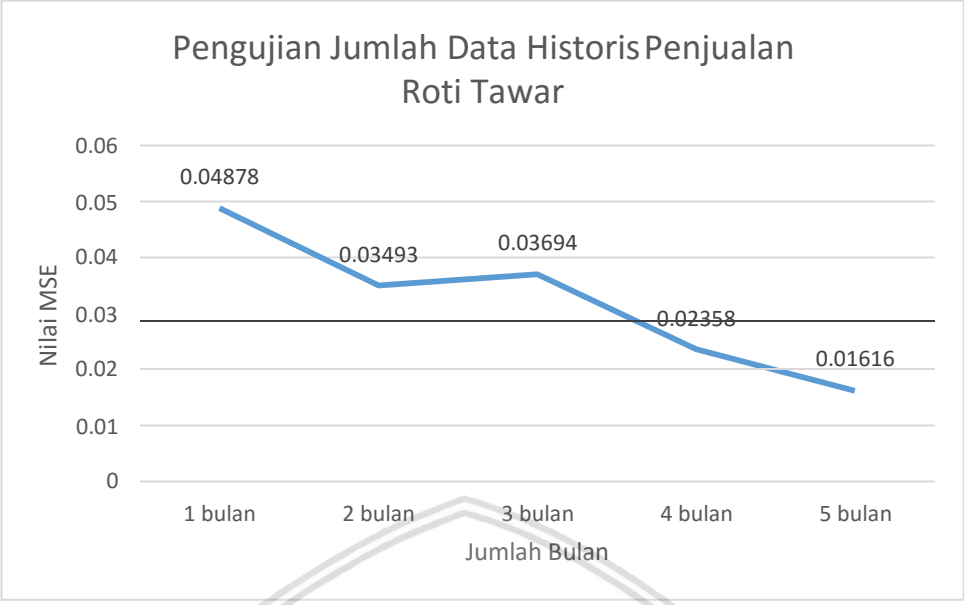
6.3 Pengujian Jumlah Data Historis Penjualan

Pengujian ini dilakukan untuk mencari nilai MSE yang paling rendah atau yang terkecil. Pengujian ini adalah pengujian jumlah data historis penjualan. Pengujian ini menguji jumlah data mulai dari pengujian menggunakan data 1 bulan saja sampai data historis penjualan selama 5 bulan. Jumlah *neouron* pada *hidden layer* pada pengujian ini adalah menggunakan jumlah *neuron* yang memiliki nilai MSE terkecil yang didapatkan pada pengujian jumlah *neuron* pada *hidden layer* yaitu menggunakan 7 *neuron* untuk roti tawar dan cake, dan 2 *neuron* untuk roti manis. Peneitian ini menggunakan jumlah fitur yang memiliki nilai MSE terkecil pada pengujian sebelumnya yaitu pengujian jumlah fitur data historis penjualan yaitu sebanyak 4 fitur untuk roti tawar dan cake, dan 5 fitur untuk roti manis. Maksud dari 4 fitur ini adalah fitur data historis penjualan selama 4 hari keblakang. Jadi pengujian ini menggunakan data hari h-4 sampai hari h-1 hari sebelum target prediksi. Pengujian ini juga menggunakan fungsi aktivasi sigmoid sebagai fungsi aktivasinya. Perbandingan data *training* dan data *testing* adalah 80% dan 20%. Setiap pengujian jumlah data historis penjualan dilakukan sebanyak 10 kali. Setiap kali melakukan percobaan, sistem akan menampilkan nilai MSE nya, kemudian dari 10 kali percobaan itu nilai MSE nya dirata-rata dan kemudian nilai rata-rata MSE tersebut di bandingkan dengan pengujian jumlah data historis penjualan yang lain. Hasil pengujian dari pengujian jumlah data

historis penjualan untuk masing-masing roti akan ditunjukkan pada Tabel 6.7, Tabel 6.8, dan Tabel 6.9. Sedangkan untuk grafik hasil pengujian jumlah data historis penjualan ditunjukkan pada Gambar 6.7, Gambar 6.8, dan Gambar 6.9.

Tabel 6.7 Hasil Pengujian Jumlah Data Historis Penjualan Roti Tawar

Percobaan ke-*i*	**Nilai MSE pada Jumlah Data *Training***				
	1 Bulan	**2 Bulan**	**3 Bulan**	**4 Bulan**	**5 Bulan**
1	0.1726	0.0387	0.0328	0.0231	0.0184
2	0.0618	0.0354	0.0379	0.0236	0.0165
3	0.0202	0.0372	0.0326	0.0259	0.0155
4	0.0374	0.0363	0.0397	0.0235	0.0152
5	0.0345	0.0291	0.0336	0.0213	0.0152
6	0.059	0.0399	0.0383	0.026	0.0143
7	0.0266	0.0254	0.0356	0.0214	0.0183
8	0.0391	0.0376	0.0451	0.0236	0.0163
9	0.0227	0.0338	0.0385	0.0231	0.0165
10	0.0139	0.0359	0.0353	0.0243	0.0154
Rata-rata nilai MSE	0.04878	0.03493	0.03694	0.02358	0.01616



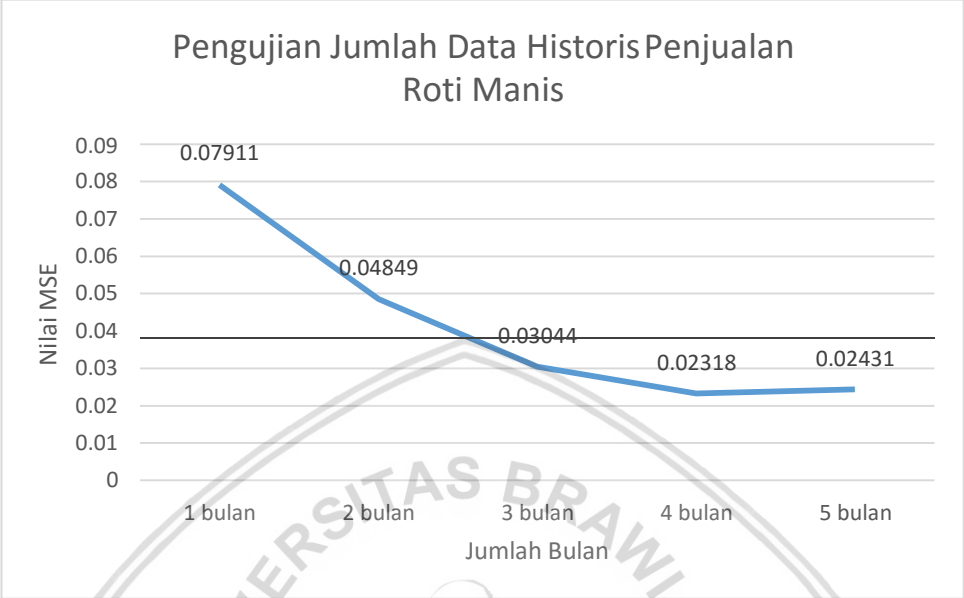
Gambar 6.7 Hasil Pengujian Jumlah Data Historis Penjualan Roti Tawar

Sesuai dengan tabel 6.7 dan grafik pada gambar 6.7, nilai MSE terbaik pada roti tawar dihasilkan dengan menggunakan 5 bulan data dengan MSE sebesar 0,01616.

Tabel 6.8 Hasil Pengujian Jumlah Data Historis Penjualan Roti Manis

Percobaan ke-*i*	Nilai MSE pada Jumlah Data *Training*				
	1 Bulan	2 Bulan	3 Bulan	4 Bulan	5 Bulan
1	0.0673	0.0539	0.027	0.0204	0.0241
2	0.0513	0.0446	0.028	0.0236	0.0275
3	0.0686	0.0477	0.0313	0.0223	0.0268
4	0.0913	0.0429	0.0337	0.0182	0.0249
5	0.0692	0.0436	0.0288	0.0214	0.0236
6	0.0687	0.0381	0.0319	0.0271	0.0239
7	0.1742	0.0427	0.0304	0.0222	0.0215
8	0.054	0.0583	0.0323	0.0299	0.0262
9	0.0831	0.0674	0.0252	0.0248	0.0207
10	0.0634	0.0457	0.0358	0.0219	0.0239

Rata-rata nilai MSE	0.07911	0.04849	0.03044	0.02318	0.02431



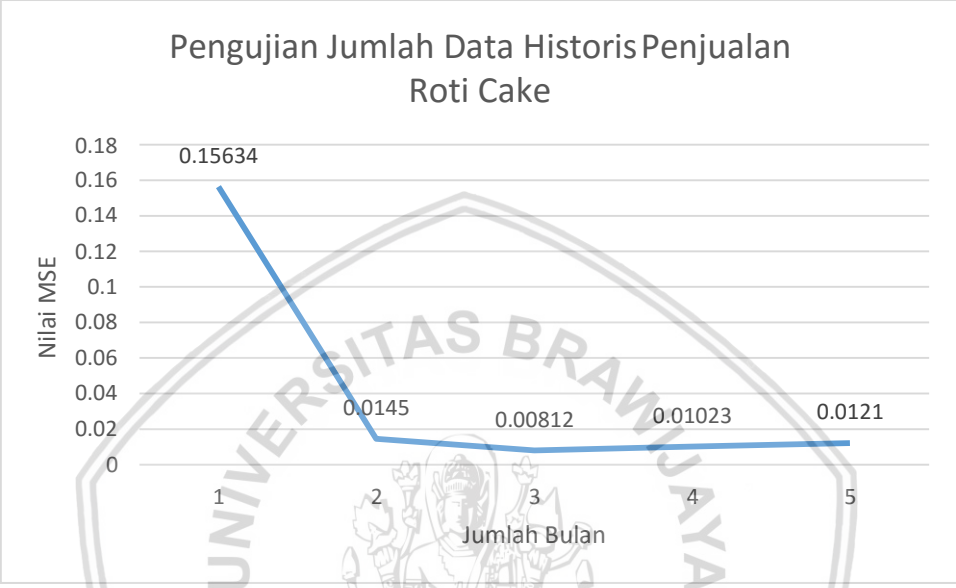
Gambar 6.8 Hasil Pengujian Jumlah Data Historis Penjualan Roti Manis

Sesuai dengan tabel 6.8 dan grafik pada gambar 6.8, nilai MSE terbaik pada roti manis dihasilkan dengan menggunakan 4 bulan data dengan MSE sebesar 0,02318.

Tabel 6.9 Hasil Pengujian Jumlah Data Historis Penjualan Roti Cake

Percobaan ke-*i*	**Nilai MSE pada Jumlah Data *Training***				
	1 Bulan	**2 Bulan**	**3 Bulan**	**4 Bulan**	**5 Bulan**
1	0.2101	0.0156	0.0085	0.0096	0.0114
2	0.139	0.0161	0.008	0.0094	0.0118
3	0.1508	0.0168	0.0083	0.009	0.0145
4	0.3103	0.0155	0.0077	0.0103	0.0143
5	0.1669	0.0125	0.0082	0.0089	0.0112
6	0.0995	0.0163	0.0074	0.0127	0.0117
7	0.2421	0.0148	0.0081	0.0123	0.0113
8	0.0257	0.0115	0.0087	0.0076	0.0126

9	0.0415	0.0138	0.0084	0.0127	0.0106
10	0.1775	0.0121	0.0079	0.0098	0.0116
Rata-rata nilai MSE	0.15634	0.0145	0.00812	0.01023	0.0121



Gambar 6.9 Hasil Pengujian Jumlah Data Historis Penjualan Roti Cake

Sesuai dengan tabel 6.9 dan grafik pada gambar 6.9, nilai MSE terbaik pada roti cake dihasilkan dengan menggunakan 3 bulan data dengan MSE sebesar 0,00812. Dapat disimpulkan bahwa jika menggunakan data yang lebih banyak, maka jaringan akan semakin kuat dan memiliki hasil prediksi yang lebih bagus karena sistem dapat melakukan proses *learning* lebih banyak jika sistem menerima data yang banyak pula. Namun, tidak menjamin 100% mendapatkan grafik nilai MSE yang mengecil dengan konstan jika data diperbanyak. Hal itu dikarenakan ada faktor lain yaitu data yang memiliki tingkat variasi tinggi yang menyebabkan semakin banyak data maka memungkinkan untuk mendapatkan nilai MSE yang lebih besar dan semakin tidak konstan penurunan nilai MSEnya

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang dapat kita peroleh dari penelitian yang melalui proses pengujian dan pembahasan dari prediksi penjualan roti menggunakan metode *Extreme Learning Machine*(ELM) pada Harum *Bakery* ini adalah :

1. Metode *Extreme Learning machine* (ELM) adalah metode pembelajaran yang memiliki beberapa langkah untuk menghasilkan suatu prediksian. Yang membedakan ELM dengan metode *learning* lain adalah, ELM menggunakan nilai bias dan *input weight* dengan nilai random. ELM merupakan metode yang memiliki kecepatan yang tinggi dalam melakukan *learning* dibandingkan dengan metode lain. Pada penelitian ini, peneliti mengimplementasikan metode ELM kedalam sistem aplikasi berbasis android, dimana pada aplikasi ini terdapat beberapa tampilan antarmuka yakni seperti halaman import data yang merupakan halaman awal jika aplikasi dibuka. Halaman ini adalah halaman dimana pengguna harus memasukkan data seperti *hidden layer,* input data historis penjualan, dan variasi fitur data. Halaman berikutnya adalah halaman normalisasi. Halaman ini berisi data awal yang telah dinormalisasi menggunakan *min max normalization*. Halaman berikutnya adalah halaman *training* yang berisi proses-proses dan hasil *training* data. Kemudian ada halaman *testing* yang berisi proses-proses dan hasil *testing* data. Kemudian ada halaman denormalisasi yang berisi hasil dari proses denormalisasi data. Halaman selanjutnya adalah halaman evaluasi yang berfungsi untuk menampilkan nilai erornya enggunakan metode MSE. Dan yang terakhir adalah halaman *weight &* bias yang berisi nilai bias dan *input weight* yang digunakan pada implementasi sistem aplikasi prediksian ini. Terdapat langkah-langkah pada aplikasi ini untuk melakukan suatu prediksian menggunakan metode ELM dimulai dari memasukkan data historis penjualan roti tawar, roti manis, dan cake. Kemudian memasukkan jumlah variasi data historis penjualan ketiga jenis roti, dan memasukkan jumlah *neuron* pada *hidden layer*. Perhitungan dimulai dari normalisasi data, yang kemudian setelah dinormalisasi data dibagi menjadi dua, data pertama sebesar 80% untuk data *training,* dan 20% data sisanya adalah untuk data *testing.* Setelah itu, akan ada 2 proses, yaitu proses *training* dan proses *testing*. Proses *training* memiiki beberapa langkah yaitu pertama menghitung hasil output *hidden l*ayer dengan fungsi aktivasi, dan kemudian setelah melalui beberapa perhitungan, proses *training* ini menghitung nilai *output weight* yang kemudian *output weight* tersebut digunakan pada proses *testing*. Untuk proses *testing* itu sendiri memiliki langkah-langkah perhitungan yang hampir sama dengan proses *testing*. Prosesnya adalah dengan

menghitung output *hidden layer* dengan fungsi aktivasi, setelah itu, menghitung nilai keluaran *output layer,* dan hasil dari proses *testing* ini menghasilkan suatu prediksian yang kemudian hasil ini dikembalikan nilainya ke nilai normal dengan metode denormalisasi. Untuk hasil MSE bisa kita dapat dari proses perhitungan *output layer*.

2. Hasil dari pembahasan dan pengujian yang diukur menggunakan metode *Mean Square Error* (MSE) pada prediksi penjualan roti pada Harum *Bakery* adalah :

   a. Pada pengujian dan pembahasan jumlah *neuron* pada *hidden layer* didapatkan nilai rata-rata MSE terkecil pada jumlah *neuron* sebanyak 7 dimana nilai MSEnya sebesar 0.03694 pada roti tawar dan 0.00812 pada roti cake. Sedangkan roti manis menghasilkan nilai MSE terkecil pada pengujian menggunakan 2 *neuron* sebesar 0.03153. Dapat ditarik kesimpulan bahwa semakin banyak jumlah *neuron* maka tidak menjamin hasil dari prediksian mendapat nilai MSE yang lebih kecil. Hal ini dikarenakan data yang memiliki variasi yang tinggi. Kemudian, jika dilihat pada tabel pengujian, terdapat angka yang kecil sekali kemungkinannya meghasilkan nilai MSE yang sama dalam suatu pengujian dengan jumlah *neuron* yang sama. Hal ini dikarenakan nilai bias dan *input weight* yang dirandom setiap kali proses prediksian. Sehingga kita perlu melakukan beberapakali pengujian dengan parameter yang sama berkali-kali demi mendapatkan hasil yang lebih optimal.

   b. Pada pengujian dan pembahasan jumlah fitur data historis penjualan didapatkan nilai rata-rata MSE terkecil pada jumlah fitur sebanyak 4 dimana nilai MSEnya sebesar 0.03694 pada roti tawardan 0.00812 pada roti cake. Dan roti manis menghasilkan nilai MSE terkecil pada pengujian menggunakan 5 jumlah fitur data dengan nilai MSE sebesar 0.0304. Dapat ditarik kesimpulan bahwa semakin sedikit fitur data historis penjualan roti, besar kemungkinan semakin akurat hasil yang diperoleh. Hal ini dikarenakan, semakin sedikit fitur maka semakin banyak jumlah data training dan testing dan hal itu membuat hasil prediksi yang lebih baik. Kemudian, jika dilihat pada tabel pengujian, terdapat angka yang kecil sekali kemungkinannya meghasilkan nilai MSE yang sama dalam suatu pengujian dengan jumlah *neuron* yang sama. Hal ini dikarenakan nilai bias dan *input weight* yang dirandom setiap kali proses prediksian. Sehingga kita perlu melakukan beberapakali pengujian dengan parameter yang sama berkali-kali demi mendapatkan hasil yang lebih optimal.

   c. Pada pengujian dan pembahasan jumlah fitur data historis penjualan didapatkan nilai rata-rata MSE terkecil pada jumlah data historis penjualan sebanyak 5 pada roti manis yaitu sebesar 0.01616, 4 pada roti manis yaitu sebesar 0.02839, dan 3 pada roti cake yaitu sebesar

0.00928. Sesuai dengan penelitian yang peneliti lakukan, dapat ditarik kesimpulan bahwa tidak menjamin 100% semakin banyak jumlah data penjualan menghasilkan nilai MSE yang semakin kecil, hal ini dikarenakan tidak semua data memiliki variasi nilai yang bagus atau *noise* yang kecil. Sehingga dapat ditarik kesimpulan bahwa semakin banyak jumlah data yang digunakan, maka tidak menjamin nilai MSE dari prediksian membaik. Kemudian, jika dilihat pada tabel pengujian, terdapat angka yang kecil sekali kemungkinannya meghasilkan nilai MSE yang sama dalam suatu pengujian dengan jumlah *neuron* yang sama. Hal ini dikarenakan nilai bias dan *input weight* yang dirandom setiap kali proses prediksian. Sehingga kita perlu melakukan beberapakali pengujian dengan parameter yang sama berkali-kali demi mendapatkan hasil yang lebih optimal.

7.2 Saran

Saran untuk pengembangan dari penelitian yang berjudul Implementasi metode *Extreme Learning Machine* (ELM) (Study Kasus : Harum Bakery) adalah :

1. Peneliti selanjutnya dapat memperbaiki kelemahan yang ada pada penelitian ini yaitu mengatasi variasi data yang tinggi atau *noise* yang tinggi dan menentukan jumlah neuron terbaik untuk mengoptimasikan prediksian dengan cara memodifikasi metode ELM. Pemodifikasian ELM ini bermacam macam, bisa dimodifikasi seperti dengan metode *Differential Evolution* (DE-ELM), *Optimally Prunded* (OP-ELM), Structural Risk Minimization (SRM-ELM), dan lain lain. Menggabungan ELM dengan metode lain menghasilkan nilai prediksian yang lebih baik, tetapi memiliki waktu komputasi yang lebih lama jika dibandingkan dengan metode ELM. Selain itu, karena nilai bias dan *input weight* didapatkan secara random persekali predikisian, untuk mendapatkan nilai MSE yang optimal, proses prediksi bisa dilakukan berulang-ulang sehingga mendapat nilai rata-rata MSE yang optimal.

DAFTAR PUSTAKA

Arestyani, M. (2017). *Klasifikasi Penyimpangan Tumbuh Kembang Anak Menggunakan Metode Extreme Learning Machine (ELM).* Malang, Jawa Timur, Indonesia: Universitas Brawijaya.

Fardani, D., Wuryanto, E., & Werdiningsih , I. (2015). Sistem Pendukung Keputusan Peramalan Jumlah Kunjungan Pasien Menggunakan Metode : Extreme Learning Machine (Studi Kasus : Poli Gigi RSU Dr. Wahidin Sudiro Husodo Mojokerto). *Journal of Information System Engineering and Business Intelegence , 1*.

Fikriya, A., Irawan, M., & Soetrisno. (2017). Implementasu Extreme Learning Machine Untuk Pengenalan Objek Citra DIgital. *Jurnal Sains Dan Seni ITS , 6*.

Gusti, A. (2017). *Prediksi Penjualan Mi Menggunakan Metode Extreme Learning Machine (ELM) Di Kober Mi Setan Cabang Soekarno Hatta.* Malang, Jawa Timur, Indonesia: Universitas Brawijaya.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme Learning Machine Theory And Applications. *Neurocomputing* , 489-501.

Huang, G., Zhu , Q., & Siew, C. (2004). Extreme Learning Machine : a New Learning Scheme of Feedforward Neuralnetworks. *IJCNN* , 25-29.

Humaini, Q. (2015). *Jaringan Syaraf Tiruan Extreme Learning Machine (ELM) Untuk Memprediksi Kondisi Cuaca Di Wilayah Kota Malang.* Malang, Jawa Timur, Indonesia: Universitas Islam Negri Maulana Malik Ibrahim.

Istiqara, K. (2017). *Prediksi Kebutuhan Air PDAM Kota Malang Menggunakan Metode Fuzzy Time Series Dengan Algoritma Genetika.* Malang, Jawa Timur, Indonesia: Universitas Brawijaya.

Jain, Y., & Bhandare, S. (2011). Min Max Normalization Based Data Perturbation Method for Privacy Protection. *Int. J. Comput* , 45-50.

Kusumadewi, S. (2003). *Artificial Inteligence (Teknik dan Aplikasinya).* Yogyakarta, Jawa Tengah, Indonesia: Graha Ilmu.

Mate, A., Ferrandez, A., & Pereal, J. (2016). Hybird Intergerated Architecture For Energy Consumption Prediction. *Elsevier , 63*, 131-147.

Perdana, I. (2016). *Simulasi dan Prediksi Jumlah Penjualan Air Menggunakan Jaringan Syaraf Tiruan Backpropagation (Studi Kasus : PDAM Tirta Kepri).* Tanjungpinang: Universitas Maritim raja Ali Haji.

Ramadhanty, A. (2016). *Prediksi Jumlah Permintaan Koran Menggunakan Metode Extreme Learning Machine.* Malang, Jawa Timur, Indonesia: Universitas Brawijaya.

Sinuhaji, F. (2009). *Jaringan Syaraf Tiruan Untuk Prediksi Keputusan Medis Pada Penyakit Asma.* Medan, Sumatera Utara, Indonesia: Universitas Sumatera Utara.

Siwi, I., Cholissodin, I., & Furqon, M. (2016). Peramalan Produksi Gula PasirMenggunakan Extreme Learning Machine (ELM) Pada PG Candi Baru Sidoarjo. *DORO : Repositori Jurnal Mahasiswa PTIIK , 8*.

Sukmarini, Y., Statiswaty, & Ramadhan, R. (2016). Penerapan Metode Exponential Smoothing pada Peramalan Penjualan Dalam Penentuan Kuantitas Produksi Roti (Studi Kasus Perusahaan Roti Dhiba Kendari). *semanTIK , 2*, 229-236.

Triyono, G. (2011). Pertimbangan Melakukan Denormalisasi Pada Model Basis Data Relasi. *Jurnal TELEMATIKA MKOM , 3*, 19-25.